Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER IN COMPUTER SCIENCE

A software for agricultural researchers in design and analysis of experiments

Thainese, J.

Award date:
1990

Awarding institution:
University of Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. May. 2023

Année académique 1989 -1990.

A SOFTWARE FOR AGRICULTURAL RESEARCHERS IN DESIGN AND ANALYSIS OF EXPERIMENTS

Mémoire de fin d'études en vue de l'obtention du diplôme
de licence et maîtrise en informatique.

par J. THAINESE sj

Promoteur : Mme M. NOIRHOMME-FRAITURE

ACKNOWLEDGEMENT

I thank wholeheartedly Professor M. Noirhomme for her encouragement and valuable guidance. My deep gratitude to Professor J.L. Hainaut for his expert help in designing the data base. I am thankful to O. Marchand for his help in implementing the data base design in NDBS. I am grateful to Professor J.J. Claustriaux of the Faculty of Agronomical Sciences, Gembloux, for helping me to formulate the objective of this dissertation and for giving practical suggestions.

My profound gratitude to my jesuit community at rue Bayar, Namur, which had been very close to me in my life and work and made it possible to accomplish this task. My sincere thanks to all who had been in one way or another helpful to achieve this goal.

Above all I thank God for his goodness and love towards me.

CONTENTS

CHAPTER 1

INTRODUCTION

The heart of India beats primarily in its 600,000 villages and agriculture is a mainstay of the majority of Indian masses. The exploding population rate and poverty grimly dictate India's requirements for essential commodities like food, clothes and shelter. Therefore, any genuine effort towards development should be based on and centered around rural and agricultural growth. Increasing the quantity and quality of agricultural production is India's primarily problem and hence agriculture has been given the high priority in Five-year plans. Government and private institutions have set up many Research stations in agriculture. The major thrust of agricultural scientists at present in India is to develop improved varieties and the associated technology for small farmers in regions of adverse ecological conditions, to help produce more from limited land and resources.

Agricultural experiments of this nature require enormous time and effort in planning, design layout in the field and analysis of data. This is my personal experience too as a student of MSc Statistics in conducting varietal, spacing and manurial experiment in paddy cultivation using a 3x2x2 Mixed factorial design in Madras Christian College, Tambaram (India) in 1976 and later as a professor of statistics for BSc (Farm Science and Rural Development) students in Loyola Academy in Osmania University, Hyderabad (India). It is a formidable task for the agricultural research workers to store and retreive the data and information of various experiments of many years in a meaningful way and to compare, analyse and make valid conclusions from them. Another important role of the researcher is to represent the valid conclusions in a simple and comprehensible way.

The growth in the computer industry has made it possible to store, retreive and process vast amounts of information in the computer. The advent of microcomputers has paved the way for easy displacement of computers to the working site in the fields spread out in different regions. Therefore an adequate need for software development for microcomputers to help the researchers in their design layout, storage, retreival, analysis of agricultural information. The automation of every face of the activity, of the agricultural research workers, with man and machine interface, would be of immense help for them.

Taking into account the problem faced in the domain of agriculture as described above and the advancement of computer aids, the **main objective** of this work will be the following: to develop a software system for microcomputers to help the agricultural

research workers involved in cultivation of different crops with statistical technique of design layout in the field, analysis of observed data and simple and comprehensible representation of the results obtained. It is devised for application in India using the data collected there, but the ultimate aim is that it could be used in any country. It is envisaged for statistical designs such as Randomized Complete Block Design, 3x2x2 Mixed Factorial Design (with Randomized Block Design layout) and Split-Plot design (without confounding). Application of the above three statistical designs will be for one observation for each experimental plot.

CHAPTER 2

SYSTEM ARCHITECTURE

In Jackson's view, perhaps the most critical factor in determining the life cycle cost of a program is the degree to which it faithfully models the problem environment, that is, the degree to which the program model corresponds to the real world (JACKSON, 75). Our program here is devised to correspond to the real world of agricultural researchers in India, but it must carry with it the essential qualities of a software, namely: 1. generality, i.e. great independence from the particularity of Indian situation 2. flexibility, i.e. easy adaptation to the needs of other situations in other countries and 3. extensibility, i.e. automation of new functions in relation with those already realized, without any major changes to the earlier.

For our design strategy, we use the functional decomposition method, employing the "divide and conquer" technique applied to programming. From the need of the agricultural research workers and the objective expressed in the last section we could decompose the real environment into the following functions, namely design layout, statistical analysis, retrieval of information about the experiments and updating of information. These are the functions which reflect the real needs and if we transform each function to fit into four modules, then these modules will be close to the domain of application. Of course these modules depend on other modules to fulfill their functions, namely, input/output of data, managing the data base primitives of retreival, update of data and representation of the information, statistical layout and statistical analysis on the screen using colours, graphics, visual aids.

Therefore the next level of modules will be:

1. Input / Output modules
2. Graphic management
3. Data base management module

The next level will be the system and tools necessary for the above levels. Since the modules of this level, namely terminal tool, graphic tool, DBMS and the lower levels are readily available, our concern will go to the upper levels.

On the top of all the modules we add one more module called coordinator which will help coordinate the application functions. The hierarchical structure of the modules proposed above is presented in the figure 2.1.

This kind of well defined segmentation of the project effort ensures the system modularity. Each task forms a separate, distinct program module. This method improves the flexibility, comprehensibility and extensibility of the system. System errors and

deficiencies can be traced to specific system modules, thus limiting the scope of detailed error searching. The functional decomposition helps us for the construction of a good hierarchical program structure.

The above hierarchical structure using the logical relation "use" makes easy the development (specifications, conceptions, validations and documentations) and maintenance of the software. The relation "import/export" helps the user to follow the different stages of the operations and supply the information needed to proceed further.

The relation "use" can be described as follows: If A and B are two modules, then A uses B and B does not use A. The validity of A depends on the availability of the correct version of B. During conception, the specifications of B can be developed without any concern about the specifications of A. The relation "use" is very often implemented by "call" in the physical architecture.

The relation "import/export" can be described in the following way: Let us suppose that A and B are two modules. A exports information to B and B imports information from A only if A supplies the information that is needed by B. The converse is also true.

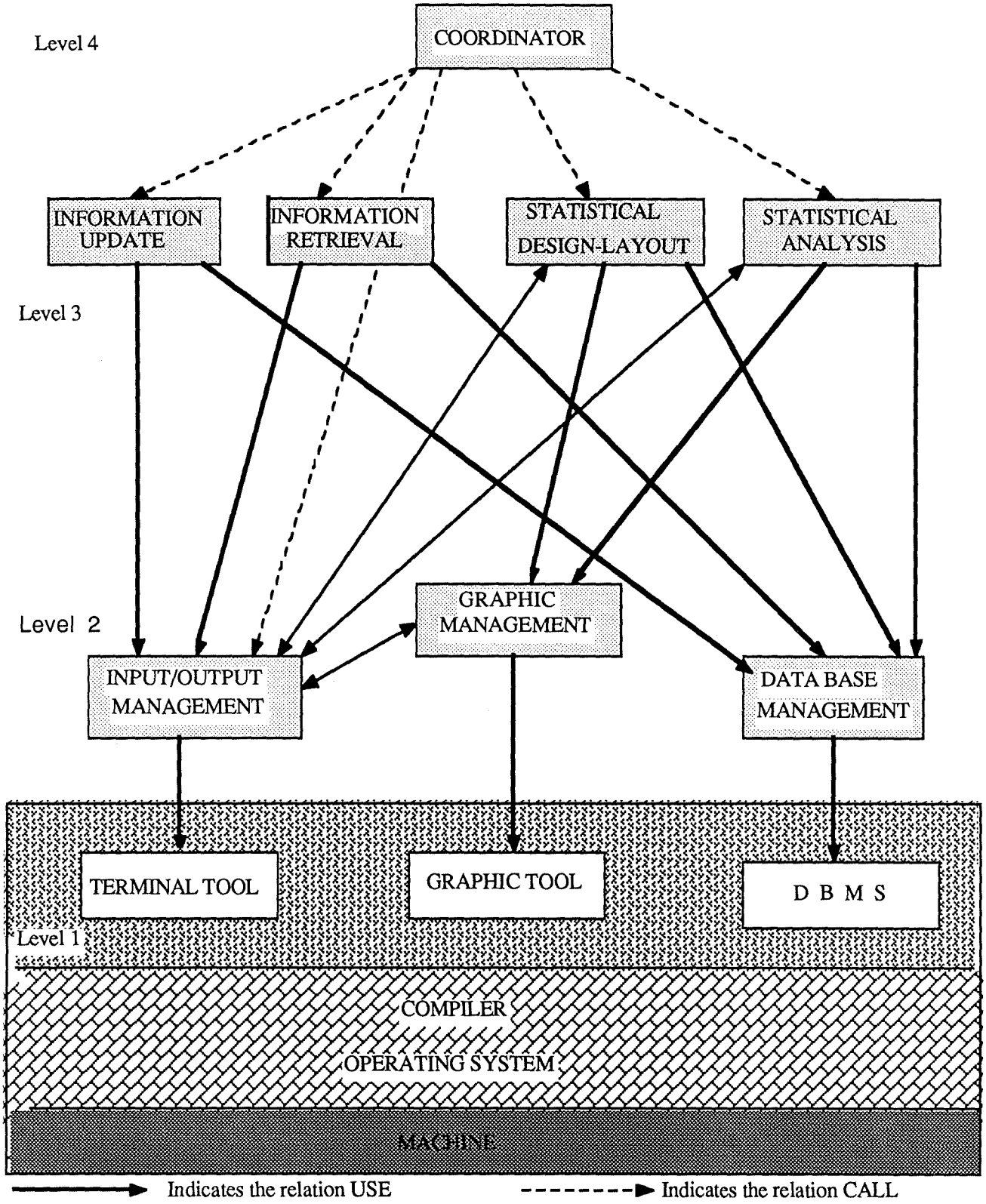


Figure : 2.1 LOGICAL-ARCHITECTURE

CHAPTER 3

DATA BASE DESIGN

In the logical architecture, the Data Base Management Module should represent the real environment of the statistical experiments so that the evolution and dynamism of the real world can be integrated into the development of the software program. Since all the higher level modules of the functions heavily depend on this module, it becomes the kernel of the whole software system. Hence we first design a data base in order to develop the data base management module and all other modules. The construction of the data base design will be as follows :

3.1. Introduction to NDBS
3.2. Models and Methods
3.3. Data Base Design

3.1. Introduction to NDBS (HAINAUT, 87)

Since the option to implement the software is in PASCAL language and that too in microcomputers, it has been decided to implement the data base management module in NDBS (Network Data Base System).

NDBS is a comprehensive data base environment for the rapid development of complex data base applications in PASCAL on microcomputers. NDBS is a member of Network and Entity-Relationship DBMSs. It has been designed and developed at the "Institut d'Informatique" of the University of Namur, Belgium. Data model for NDBS is derived from the current Entity/Relationship approach. It is simple, powerful and natural. A clear distinction is established between the semantic structure of the data and the technical parameters. The latter can always be ignored. The programming interface which is given to the programmer is much simpler than the current state in that domain. The most complex programs can be written by using 12 data base functions only. Moreover, the functions offer a regular interface totally complying with the PASCAL data types and programming rules.

NDBS (version 1) includes three main components: the **data base handler** is a set of procedures that give the programmer the data base function to access a data base to find data according to several criteria and to update the data. The **schema processor** is a program that allows the user to give the description of data bases, to modify, display and

print data base descriptions, to generate operational data base from the description. The **high-level language pre-processor** allows the programmer to write data base programs in a high-level language called ADL-PASCAL (standing for Access Algorithm Description Language embedded in PASCAL). The pre-processor transforms an ADL-PASCAL program into a pure PASCAL program making use of the basic data base interface. The pre-processor is a NDBS program that uses the data dictionary.

3.2. Models and Methods

3.2.1. Entity-Relationship model[1]

A data base is a collection of data that describe a subset of the real world. We may call this subset application domain or application environment. The structure of this application domain is based on the Entity-Relationship modeling approach that is standard in the data base design and in the data base description. The Entity-Relationship approach is simple and powerful enough to be adopted for designing a very large and complex data base.

The Entity-Relationship model is a model that allows the data base designer to express the semantics of the information of the application domain with the help of three concepts, namely, entities, attributes and inter-entity-relationships.

An **entity** is an object or concept meaningful to the organization about which there is a need to record data. In a banking environment examples of entities are CUSTOMER, BANK ACCOUNT and so on. In a warehouse the entities are SUPPLIER, PART, SHIPMENT and the like. In the university the entities are STUDENT, STAFF, DEGREE, COURSE and so on. Entities are classified as **entity types**. For instance, STUDENT is a collection of all students. The collection of the students is constantly changing, but the notion of STUDENT entity type is a stable, static concept for the university environment.

Every entity has some basic **attributes** that characterize it. A house can be described by its size, colour, age and surroundings. A customer of a bank may be described by such attributes as customer number, customer name and customer address. We associate the attributes NUMBER, NAME and ADDRESS with the CUSTOMER entity type. Some attributes can play a special role for the entity type. No two customers have the same number. We can call that NUMBER as the **identifier** of the CUSTOMER entity type.

---

[1]The concepts and ideas that follow are taken from HAINAUT, 87.

Through the customer identification number, we can determine the name and address of the person in question.

In general, entities are not isolated in the application domain. Each student belongs to a university, each employee belongs to a department, in a hospital environment a patient is assigned to a bed and in a bank an account is held by a customer. These situations correspond to **relationship** between entities. A relationship is a mapping or linkage between two entities. The collection of similar relationships which bear the same meaning is called a **relationship type**. In the examples above, the linkage can be described by the BELONG-TO relationship type between EMPLOYEE and DEPARTMENT, IS-ASSIGNED-TO relationship type between PATIENT and BED, and IS-HELD-BY relationship type between ACCOUNT and CUSTOMER.

Each employee can be associated with **only one** department and each department can be associated with **any number** (say N) of employees, via the BELONG-TO relationship type. Such a relationship type is called 1-N or one-to-many, from DEPARTMENT to EMPLOYEE, and conversely N-1 or many-to-one, from EMPLOYEE to DEPARTMENT. There exists a 1-1 or one-to-one relationship between PATIENT and BED via the IS-ASSIGNED-TO relationship type. There can be a N-N or many-to-many relationship type between two entities.

3.2.2. Data Base Design Method

3.2.2.1. Introduction

The design starts with the end users' views of the organization, called the conceptual requirements. An end user is a decision maker who uses information obtained by accessing to the data base. The end users also provide data to be stored in the data base.

In considering the end users' requirements, the following trade-offs have to be taken into account (ATRE, 80) :

- It should satisfy today's needs for information.
- It should not only satisfy today's needs but also satisfy them in some reasonable time, that is, it should satisfy the performance requirements.
- It should satisfy the anticipated as well as the unanticipated requirements of the end users.
- It should be easily expandable with the reorganization and expansion of the

enterprise.

- It should be easy to modify in changing software and hardware environments.
- Once correct data are stored in the data base, they should stay correct.
- Before inserting any data in the data base, the data should be checked for validity.
- Only authorized people should have access to the data stored in the data base.

Figure 3.1 in the next page shows the steps to be taken in designing a data base, considering the above trade-offs (ATRE, 80).

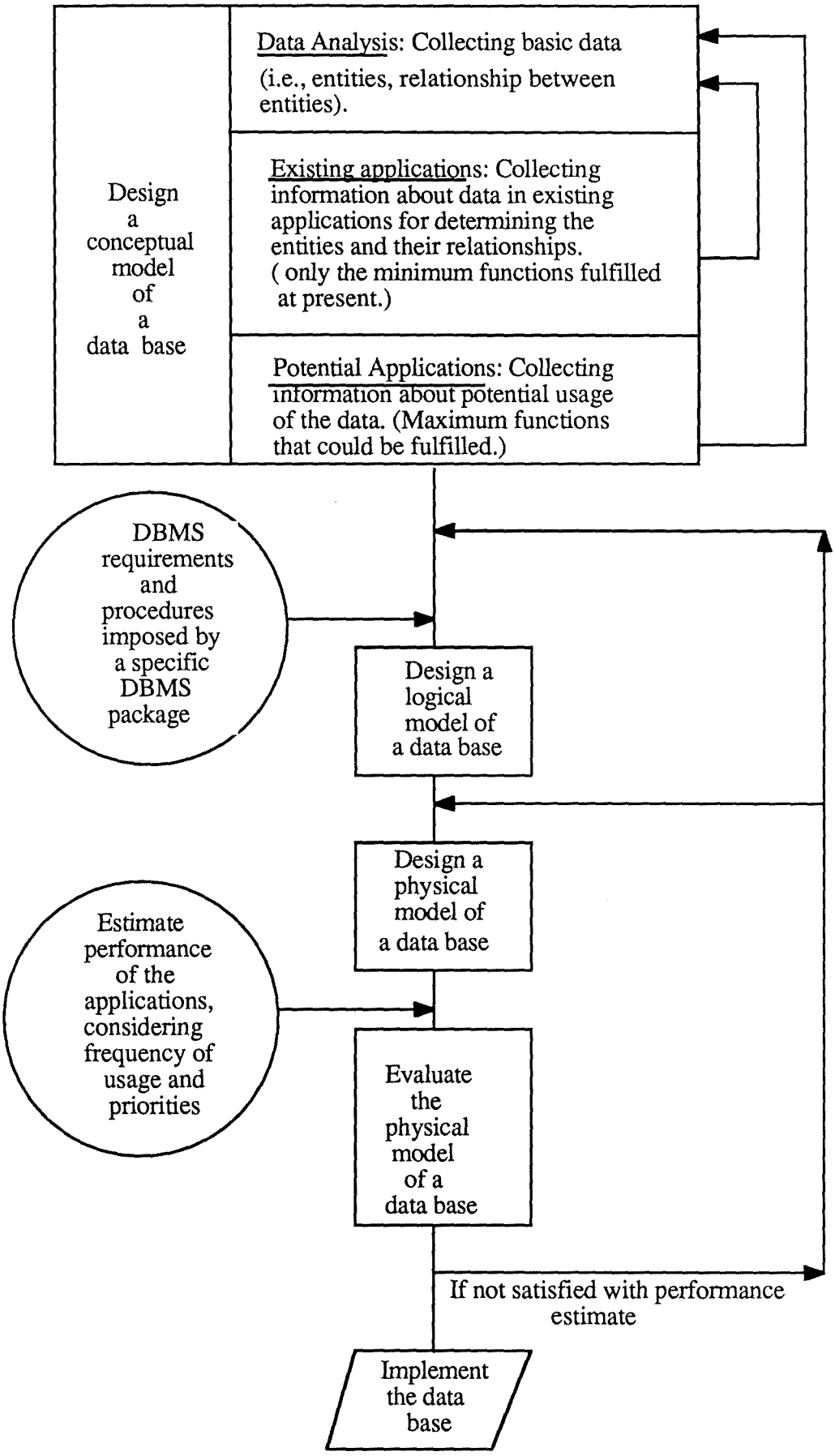


Figure 3.1 Data base design steps.

3.2.2.2. Conceptual model.

To develop the data base that satisfies today's as well as tomorrow's information needs, a conceptual model must be designed. Therefore, the data base design process starts with the conceptual requirements of a number of users. This step leads to a solution which is operational, correct and efficient (HAINAUT, 86). The conceptual model represents the entities and their relationships and is based on the data processing needs of the organizations. When determining the entities and their relationships, a data analysis is necessary. This analysis can be based on the information about data for existing, as well as for future, application. The conceptual model gives the ability to view all the data entities and their relationships to each other with no concern about their physical storage. Therefore the conceptual model is independent of individual applications, independent of the data base management system, independent of the hardware used for storing the data, and independent of the physical model of the data in the storage media (ATRE, 80).

3.2.2.3. Logical model

The conceptual model is then translated into a model compatible with the chosen data base management system. In other words, the conceptual model has to be mapped to a logical model used as an underlying structure for a data base management system. It is possible that the relationships between entities as reflected in the conceptual model are not implementable with the chosen package of the DBMS. In such situations, modifications to the conceptual model should reflect these constraints (ATRE, 80). Therefore, the version of the conceptual model that can be presented to the DBMS is called logical model.

A **NDBS data base** is organized according to the descriptive rules of the Entity-Relationship model[2]. A NDBS data base therefore contains a variable number of **entities** with their **attribute values** and their **relationships**.

The specification of the entity types, entity attributes, and relationship types of a data base is called the schema of the data base. Knowing the schema is all that is needed to write programs that use and modify data in a data base.

A NDBS data base schema accepts a simple and intuitive graphical representation. Each entity type is represented by a box containing the name of the entity type and the name of its attributes. The identifiers stand in capital letters (see figure 3.2).

2The concepts and ideas that follow are taken from HAINAUT, 87.

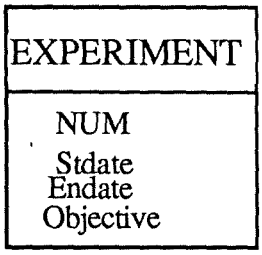


Figure 3.2: The entity type EXPERIMENT with four attributes experiment number, experiment starting date, experiment ending date and experiment objective. NUM, the experiment number, is the identifier for EXPERIMENT.

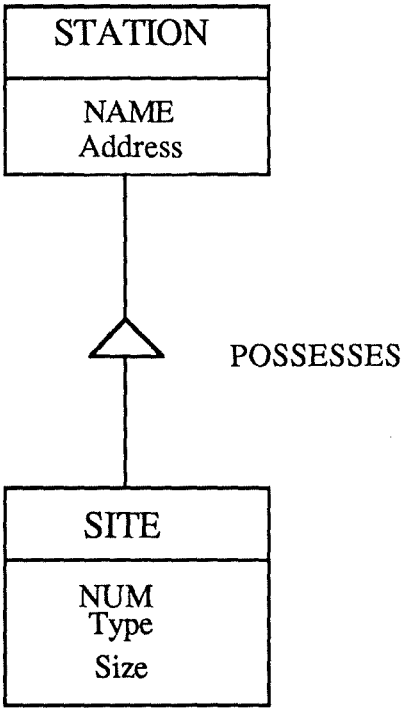


Figure 3.3: Representation of the relationship type POSSESSES between entity types STATION and SITE.

A relationship type is represented by an arc joining the boxes of two entity types (see figure 3.3). A label gives the name of the relationship type. The one-to-many (1-N) direction is indicated by a small triangle sticked on the arc.

In a program, a relationship will be used to obtain entities which are logically connected to another one. In the example of figure 3.3, once the program has got a STATION entity, it can ask for the SITE entities that are linked to it by relationship. From a SITE entity, the program can ask for the STATION entity which is connected to it. The program is said to use the path from STATION to SITE, or the path from SITE to STATION. A relationship type offers two paths. A one-to-many (1-N) path type (from STATION to SITE), and its inverse many-to-one (N-1) path type (from SITE to STATION) are associated with a relationship type. A path is directed from its **origin** entity towards its **target** entities. NDBS model does not allow direct implementation of a many-to-many (N-N) relationship. A many-to-many relationship can be transformed into two one-to-many (1-N) relationships.

A **data base** is given a name which is a valid file name in the operating system of the computer. An **entity type** has a name. An entity type name is a valid PASCAL name with a length which is one less than maximum. No two entity types of the schema may have the same name. An **entity attribute** has a name which is a valid PASCAL name. The attributes of an entity type have different names. Different entity types may have attributes with the same name. An entity type does not need to have attributes. An attribute is defined as a valid PASCAL data type. A **relationship** type has a name which is a valid PASCAL name. It is defined between two entity types. These two entity types do not need to be distinct. Such an entity type is sometimes called *recursive*.

3.2.2.4. Physical model

The physical model is a framework of the data base to be stored on physical devices. Therefore, the logical model is mapped to some physical storage such as disk, tape or drum. The physical model which takes into consideration the distribution of data, access methods, and indexing techniques, is called an **internal model**. Since a large percentage of data bases are used in an on-line environment, one has to be very much concerned with "visible" performance. One should carry out a quantitative analysis of the physical model with the average frequencies of occurrences for the different groups of the data elements, with expected auxiliary space estimates, and with the expected time estimate for retreiving and maintaining the data (ATRE, 80). Thus major flaws in the design may be detected, and the physical design of the data base can be evaluated before it is put into production.

Once a data base is installed, it is very difficult to redesign it. The concepts of physical model are useful for designing an efficient data base and when a programmer wants to foresee the performance of a given program.

While designing the physical model of a data base, we need to worry about the physical aspects of the data base, that is, the record layout on the disk, record size, buffer size and so on. The designer has to know the relationships of data elements and of entities referenced in the applications (ATRE, 80).

The physical components of a NDBS data base (HAINAUT, 87).

The internal (*physical*) representation of an entity is a byte string called the physical record of the entity, or entity record. An entity record contains the attribute values of the entity, if any. In addition, it includes technical data needed by the NDBS routines to manage the data base. Some of these data are the entity type code, sequential access pointers and path pointers.

The data are stored in an extensible data file organised in pages. A page is a fixed-length frame, typically 1 Kb long for example, in which entity records are stored, whatever their (entity) type. A page is identified by its page number. The records are stored in such a way that they do not span several pages. So, the entity record length cannot exceed the page length. One of the major factors affecting the performance of programs interacting with data base is the manner in which data are stored and accessed. In NDBS, the entities are stored in the pages according to a storage scheme, which is specific to each entity type. NDBS offers two storage schemes, namely the random storage scheme and the clustered storage scheme.

In the *random storage scheme*, the page is chosen at random in the page range, so that the records are distributed as uniformely as possible in the page range, leaving space between them. In the *clustered storage scheme*, the record is stored in the page of the last record that has been accessed to or modified. With that scheme, entity records that are created tend to be stored in the contiguous pages (clusters of records of the same type). The sequential access to them will be very efficient.

In both storage schemes, if the page chosen cannot accomodate the record, the next pages are searched for free space. If no space is found in the page range, the search goes on outside. If needed, new pages are appended to the data file. Therefore, the only limit to the extension of the data base is the disk space available.

When NDBS reads pages from a data file, it stores them in a specialised area in the main memory called the buffer, from where it extracts or modifies the entity records. The present version of the NDBS can take thirty pages into the buffer.

3.3. Data Base Design for Statistical Experiments

From my personal experience and study of statistical methods for designs of experiments and by extensive readings of the books mentioned in the foot note 3, I present now the **Description of the Environment, Conceptual Schema, Specifications, Constraints and comments, NDBS schema** and **The choice of a storage scheme.**

3.3.1. Description of the Environment[3]

An experiment is conducted, at a given time, in one or more experimental sites belonging to a station. For example, an experimental site can be a field in agricultural environment or it can be a cow-shed where cows are reared or a factory-unit in the industrial environment where each factory-unit has a certain number of machines. Each experimental site is attached to a station which has its name and address. The station has regular observations of climatic conditions, namely meteorological information. Each experiment has a particular statistical design.

In agreement with the kind of design chosen and the objective of the experiment, which involves a certain number of specific treatments, all possible or certain number of treatment combinations (in case of incomplete bloc design) are chosen by researchers in accordance with the availability of number of homogeneous experimental units and the number of treatment combinations required. Therefore, an experimental site is divided into a certain number of homogeneous blocks and each block is composed of a certain number of experimental units of optimum size. The blocks can be also called replicates. If an experimental unit is a pig, then a block can be a group of pigs of the same breed. If it is a plot of plants, then a block will be a group of homogeneous plots. If it is a machine in a factory, then a block can be a group of machines of the same make. One fixes the number of blocks and experimental units (and their sizes) using the statistical technique of replication and local-control. Each experimental unit belongs only to one particular block and a block belongs to only one experimental site, because of the characteristics of

---

[3]Sections 3.3.1., 3.3.2., 3.3.3. and 3.3.4. integrate ideas from PANSE, 67; K.MOORTY, 77 and DAGNELIE, 81.

homogeneity of a block. If this homogeneity could be found in having the same block in different experimental sites, then a block could be in different experimental sites. It could be found in factories of machines, animals of different sheds or same kind of trees of different fields as blocks. A particular experimental site may be used by many experiments at different periods of time.

In an experiment, certain treatments (or treatment combinations) are allotted to particular experimental units. The allotment is done according to the principle of randomization technique depending on the design and the objective. There are also certain treatments which are global to the experiment. For example, in agricultural experiments, the application of potash, the weeding or the irrigation could be uniform for all the plots of the experimental site. Each treatment is specified by a date and time of the application and a quantity of the treatment. An experiment may employ several treatments globally or particularly. A certain treatment may be by employed by several experiments. Certain treatments which are particular to the experiment may have a certain number of levels.

An experiment may have many observations, either global or particular to each experimental unit, e.g. the observation of the soil characteristics of an experimental field or the amount of heat produced by machines in a factory could be global observations. A particular observation may be the milk yield of a cow or the yield of paddy in a plot, depending upon the objective of the experiment. Each observation is marked by its time, its date of observation and its quantity. There may be certain experimental units with missing data.

The object of observation may be common to many experiments. An experimental unit can have many observations (e.g. milk-yield of a cow and its growth in weight). Certain observations which are particular may have a certain number of levels (e.g. growth of plants in heights observed in different periods of time; if four machines are grouped as one unit, then the observation of each machine in that unit can be considered as having a specific level, etc.). Problems (e.g. flood, cyclone, fire or disease) may affect certain experimental units in particular or it may affect globally the whole experiment. The observations collected for some particular treatments are used for statistical analysis.

3.3.2. Conceptual schema

From the description of the environment above and taking into account the present and future needs of the application, we present in figure 3.4 the conceptual schema.

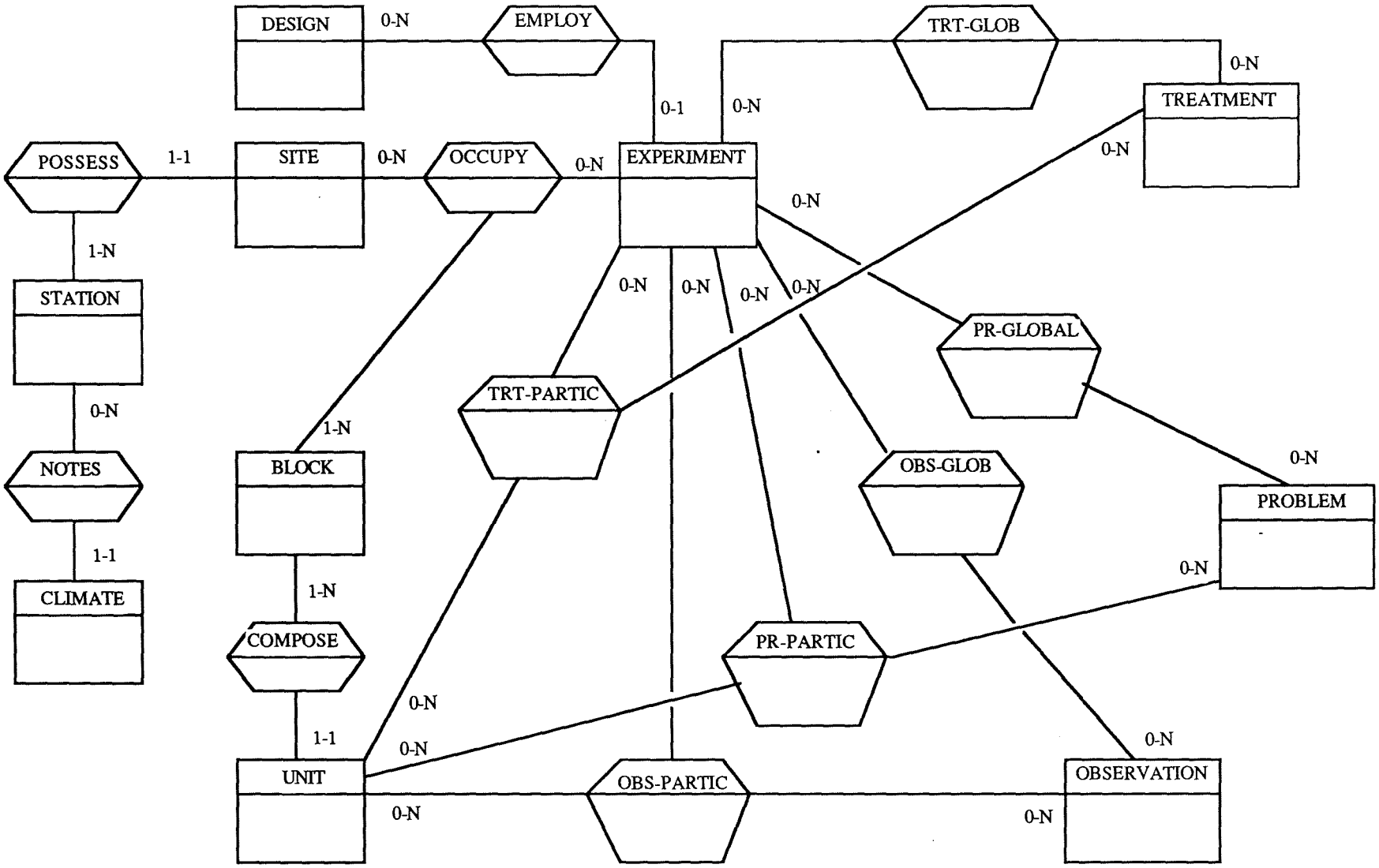


Figure 3.4: Conceptual-Schema

3.3.3. Specifications

Now we will give the **specifications** of each entity, and of each relation involved in the conceptual schema (see figure 3.4).

ENTITY:	**EXPERIMENT**	a planned set of treatments or trials carried out on some objects with a certain objective and within a given period of time, using a statistical design, which leads to a corresponding set of observations.
IDENTIFIER:	**exp.no**	Experiment-number
ATTRIBUTES:	**exp.st.date**	Experiment-starting-date
	exp.endate	Experiment-ending-date
	exp.obj	Objective of the experiment
ENTITY:	**DESIGN**	a statistical design
IDENTIFIER:	**d.name**	statistical design name (e.g. C.R.D., R.B.D., L.S.D., FACTORIAL, SPLIT-PLOT etc.)
ATTRIBUTE:	**d.desc**	Design-description (e.g. simple, with or without confounding, etc.)
ENTITY:	**SITE**	a place or material on which an experiment is conducted
IDENTIFIER:	**s.num**	Experimental-site-number
ATTRIBUTES:	**s.type**	Experimental-site-type
	s.size	Experimental-site-size
ENTITY:	**BLOCK**	a portion of the site which is likely to be more homogeneous than the whole chosen for an experiment and is composed of a certain number of experimental units
IDENTIFIER:	**b.num**	Block-number

ATTRIBUTES:	**b.rnum**	Block-row-number in the layout
	b.cnum	Block-column-number in the layout
	b.type	Block-type
ENTITY:	**UNIT**	corresponds to the smallest division of a block from which a certain number of observations is collected as the result of the treatments applied to it
IDENTIFIER:	**u.num**	Experimental-unit-number
ATTRIBUTES:	**u.rnum**	Experimental-unit-row-number in the layout
	u.cnum	Experimental-unit-column-number in the layout
ENTITY:	**STATION**	a research centre having a certain number of experimental sites and meteorological information and responsible for the day to day administration
IDENTIFIER:	**st.name**	Station-name
ATTRIBUTES:	**st.adr**	Station-address
ENTITY:	**CLIMATE**	meteorological information.
IDENTIFIER:	**cl.date**	Climate observation-date
ATTRIBUTES:	**rain**	quantity of rain in mm
	t.max	maximum temperature in °C
	t.min	minimum temperature in °C
	rad	radiation in joules/square cm
ENTITY:	**TREATMENT**	a factor chosen for planned operations on the experimental units
IDENTIFIER:	**trt.name**	Treatment-name
ENTITY:	**OBSERVATION**	a quantitative or a qualitative data collected from the units
IDENTIFIER:	**o.obj**	Observed object

ENTITY:	**PROBLEM**	an occurrence of an event that is detrimental to the experimental material for its normal functioning
IDENTIFIER:	**p.name**	Problem-name
RELATION:	**TRT-PARTIC**	Particular treatment. It is a compound relationship between the entities EXPERIMENT, UNIT and TREATMENT. It associates a treatment applied to a specific unit with an experiment.
IDENTIFIER:	**UNIT**	Entity
	EXPERIMENT	Entity
	TREATMENT	Entity
	tp.date	Date of a particular treatment
	tp.time	Time of a particular treatment
ATTRIBUTES:	**tp.qnty**	Quantity of a particular treatment
	tp.meas	Measure of a particular treatment
	tp.level	Level of a particular treatment
	tp.type	Type of a particular treatment (e.g.random, whole-plot, split-plot, etc.)
	tp.opertype	Operation-type of a particular treatment
RELATION:	**TRT-GLOBAL**	Global treatment. It is a binary relationship between the entities EXPERIMENT and TREATMENT. It associates a treatment applied uniformly to all the units with an experiment.
IDENTIFIER:	**EXPERIMENT**	Entity
	TREATMENT	Entity
	tg.date	Date of a global treatment
	tg.time	Time of a global treatment

ATTRIBUTES:	**tg.qnty**	Quantity of a global treatment
	tg.meas	Measure of a global treatment
	tg.opertype	Operation-type of a global treatment
RELATION:	**OBS-PARTIC**	Particular Observation. It is a compound relation between the entities EXPERIMENT, UNIT and OBSERVATION. It associates an observation with a specific unit and an experiment
IDENTIFIER:	**UNIT**	Entity
	EXPERIMENT	Entity
	OBSERVATION	Entity
	op.date	Date of a particular observation
	op.time	Time of a particular observation
ATTRIBUTES:	**op.qnty**	Quantity of a particular observation (missing data will have zero quantity)
	op.meas	Measure of a particular observation
	op.level	Level of a particular observation
	op.method	Method used for a particular observation
RELATION:	**OBS-GLOBAL**	Global Observation. It is a binary relation between the entities EXPERIMENT and OBSERVATION. It associates an observation made in a global way with an experiment
IDENTIFIER:	**EXPERIMENT**	Entity
	OBSERVATION	Entity
	og.date	Date of a global observation
	og.time	Time of a global observation
ATTRIBUTES:	**og.qnty**	Quantity of a global observation
	og.meas	Measure of a global observation
	og.method	Method of a global observation

RELATION:	**PR-PARTIC**	Particular Problem.It is a compound relation among the entities EXPERIMENT, UNIT and PROBLEM. It associates a problem with a specific unit and an experiment
IDENTIFIER:	**EXPERIMENT**	Entity
	PROBLEM	Entity
	UNIT	Entity
	pp.date	Date of a particular problem
	pp.time	Time of a particular problem
RELATION:	**PR-GLOBAL**	Global Problem. It is a binary relation between EXPERIMENT and PROBLEM. It associates a problem affecting globally the experimental material with an experiment
IDENTIFIER:	**EXPERIMENT**	Entity
	PROBLEM	Entity
	pg.date	Date of a global problem
	pg.time	Time of a global problem
RELATION:	**OCCUPY**	It is a compound relation between the entities EXPERIMENT, SITE and BLOCK. There is no attribute in this relation
RELATION:	**EMPLOY**	It is a binary relation between the entities EXPERIMENT and DESIGN. There is no attribute in this relation
RELATION	**POSSESS**	It is a binary relation between the entities STATION and SITE. There is no attribute in this relation
RELATION	**NOTES**	It is a binary relation beween the entities STATION and CLIMATE. There is no attribute in this relation

RELATION	**COMPOSE**	It is a binary relation between the entities BLOCK and UNIT. There is no attribute in this relation.

3.3.4. Constraints and comments on the conceptual schema

1) In the schema of the conceptual model, we have the following representation :

a) Physical Sub-System: Entities such as SITE, BLOCK, UNIT, STATION have physical characteristics.

b) Experiment: The entity EXPERIMENT and the relations TRT-PARTIC, TRT-GLOBAL, OBS-PARTIC, OBS-GLOBAL, PR-PARTIC and PR-GLOBAL have certain values, which are characteristic of each experiment. Their values are changing dynamically, e.g. the attributes DATE, TIME, QUANTITY, etc. in the relations change according to each experiment.

c) Possible Standards: Entities such as PROBLEM, TREATMENT, OBSERVATION and DESIGN are of possible standard entities for many experiments, e.g. a treatment of nitrogen can be one of the possible standard treatments among many others. A problem for a particular experiment may be one of the many possible problems.

2) The entity BLOCK has on the one hand a physical aspect and is part and parcel of an experimental SITE (experimental material) and on the other hand it has a dynamically changing character in its size, position, type, etc. depending on different experiments. Blocks are constructed under the principle of replication and local control and depending of the objective and type of experiment. Therefore, we have a triple relation OCCUPY among the entities EXPERIMENT, SITE and BLOCK. The existence of blocks is conditioned by its double link to SITE and EXPERIMENT. Through this triple relation, one can identify the following facts easily:

1. whether a BLOCK belongs to such a SITE and such an EXPERIMENT,
2. whether a SITE has been used for such an EXPERIMENT,
3. what are the different EXPERIMENTS conducted in a particular block.

3) It is possible that there exist SITES in which no experiment has been conducted so far. An experiment may be at a planning stage and not yet connected to an experimental site. And hence, no block has been formed for the experiment or in the site concerned. Therefore, the connecting link from SITE and EXPERIMENT to the BLOCK may not

exist. If there exists a site with one block, then that will signify the completely randomized design (C.R.D.) layout and the site (experimental material) concerned should be as homogeneous as necessary for such an experiment.

4) Block-row-number and block-column-number are for identifying block positions in the layout. In the same way, experimental UNIT row and column numbers are for the identification of its location in a block.

5) The size of an experimental unit can be noted in the particular observation.

3.3.5. NDBS schema

The mapping of the conceptual schema of the environment to the NDBS schema is given in the figure 3.5.

3.3.6. The choice of a storage scheme

It is very hard to find high volume transactions in the above environment. Deletion of records are rare and all the information stored has to be preserved for many years for comparison of different experiments. Therefore, the volume increases year after year. In the *random storage scheme*, an explicit page range is mandatory and the records are distributed as uniformly as possible in the page range, leaving space between them. In the long run the space between the records may not be fully utilised. Moreover, physical access to the pages is a costly operation.

In *clustered storage scheme*, the record is stored in the page of the last record that has been accessed to or modified. With that scheme, entity records that are created in group tend to be stored in contiguous pages. For daily transactions, recently created records are necessary and it is highly likely that they are all contained in the buffer space once one of those records is called. Therefore, given the nature of the environment, the clustered storage scheme will provide a better performance. By this storage scheme, one is able to save a lot of disk space and reduce the access time. Thus, this storage scheme is more economical and efficient. The entity record size and the number of records in a page for each entity are given in the appendix.

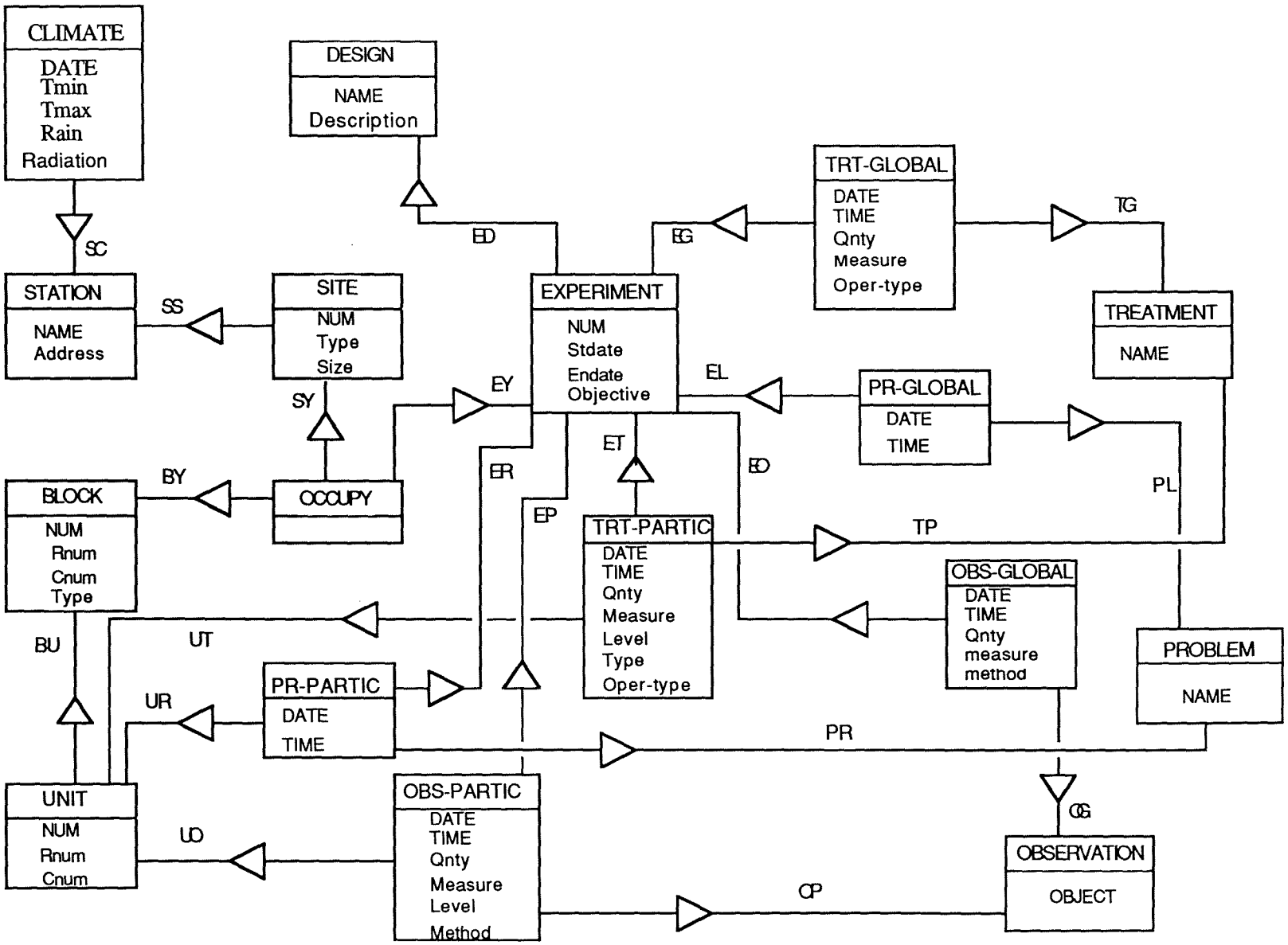


Figure 3.5: NDBS Schema

CHAPTER 4

COORDINATOR MODULE

The coordinator proposes to the user the different functions available for design and analysis of experiments in the application environment of an agricultural research centre.

The main menu represents each module as a specific function in the following way :

1. Information Update
2. Information Retrieval
3. Statistical Design Layout
4. Statistical Analysis
5. Return to the operating system

Within each of the above first four modules there are different choices of functions and subfunctions which are also equally managed by the coordinator.

4.1. In the **Information Update Module** we have the following functions :

1. Information creation
2. Information deletion
3. information modification

4.1.1. The information creation contains the following subfunctions :

1. Station creation
2. Particular treatment creation
3. Global treatment creation
4. Particular observation creation
5. Global observation creation
6. Particular problem creation
7. Global problem creation

4.1.2. The Information deletion contains the following subfunctions :

1. Block deletion
2. Particular treatment deletion
3. Global treatment deletion
4. Particular observation deletion
5. Global observation deletion
6. Particular problem deletion
7. Global problem deletion

4.1.3. The Information modification contains the following subfunctions:

1. Experiment modification
2. Particular treatment modification
3. Global treatment modification
4. Particular observation modification
5. Global observation modification
6. Particular problem modification
7. Global problem modification
8. Unit modification
9. Block modification

4.2. In the **Information Retrieval Module** we have the following functions :

1. Site occupation information
2. Experiment information
3. Particular treatment information
4. Global treatment information
5. Particular observation information
6. Global observation information
7. Particular problem information
8. Global problem information

4.3. In the **Statistical Design Layout Module** we have the following functions:

1. Designing manual layout
   - Randomized Block Design (R.B.D.) layout
   - R.B.D. Layout for Factorial Design
   - Split-Plot Design layout
2. Designing automatic layout
   - Randomized Block Design (R.B.D.) layout
   - R.B.D. Layout for Factorial Design
   - Split-Plot Design layout
3. Display layout

4.4. In the **Statistical Analysis Module** we have the following functions:

1. Randomized Block Design analysis
2. Mixed Factorial Design analysis
3. Split-Plot Design analysis

CHAPTER 5

INFORMATION UPDATE MODULE

5.1. FUNCTION: **Information creation**

In the information creation function, we present here only the following seven subfunctions: station creation, particular treatment creation, global treatment creation, particular observation creation, global observation creation, particular problem creation and global problem creation. Each of the last six subfunctions first creates the entities TREATMENT or OBSERVATION or PROBLEM, as needed for their functioning.

The description of the creation of the entities EXPERIMENT, DESIGN, BLOCK (including all the units it contains) and TRT-PARTIC involved in the *a priori* label allotments will be presented in the Statistical Design Layout Module (chapter 7). The entity CLIMATE is left for the future development.

5.1.1. SUBFUNCTION: **Station creation**

Aim: To Introduce the information concerning a new research station.

Input Parameters:

st.name : name of the station
st.adr : address of the station
ns : number of sites possessed by the station
For each site::
s.type : site type
s.size : site size

Output Messages:

One of the following messages appears :

1. 'A station with the same identifier exists already in the data base'
2. 'This station is not created in the data base'
3. 'A site belonging to this station is not created in the data base'
4. 'An access path between site and station is not connected'
5. ' This station is created in the data base'

Possible change in the data base: an occurrence of the entity STATION, with one or more occurrences of the entity SITE belonging to this station and an access path between this station and each site are created in the data base.

5.1.2. SUBFUNCTION: **Particular Treatment creation**

Aim: To introduce the information concerning a treatment applied to one or more units in particular for an experiment conducted.

Input Parameters:

exp.num : experiment number
trt.name : treatment name
u.num : unit number
tp.date : date of a particular treatment applied
tp.time : time of a particular treatment applied
tp.qnty : quantity of a particular treatment applied
tp.meas : measure used for the quantity
tp.level : if this treatment is applied as one of the factors, then the level is marked
tp.type : type of the treatment applied
tp.opertype : type of operation used to apply this treatment

Output Messages:

One of the following messages appears :
1. 'A TRT-PARTIC with the same identifier exists already'
2. 'Access path between TRT-PARTIC and EXPERIMENT not created'
3. 'Access path between TRT-PARTIC and UNIT not created'
4. 'Access path between TRT-PARTIC and TREATMENT not created'
5. 'This TRT-PARTIC is not created'
6. 'A new TRT-PARTIC is created'

Expected interaction and change in the data base : The function compares trt.name in the data base. If such a name exists, it creates only this unique TRT-PARTIC relation and connects it with the desired EXPERIMENT, UNIT and TREATMENT. If such a treatment does not exist, it first of all creates an occurrence of the TREATMENT and then introduces the new TRT-PARTIC relation verifying its constraints of uniqueness. A change in the data base is confirmed by the output message 6. This change means that an occurrence of entity TRT-PARTIC is added in the data base and an occurrence of the entity TREATMENT if necessary. The other output messages confirm that there is no change in the data base.

5.1.3. SUBFUNCTION: **Global Treatment creation**

Aim: To introduce the information concerning a treatment applied to the experimental site globally for an experiment conducted.

Input Parameters:

exp.num : experiment number
tr.t.name : treatment name
tg.date : date of a global treatment applied
tg.time : time of a global treatment applied
tg.qnty : quantity of a global treatment applied
tg.meas : measure used for the quantity
tg.opertype : type of operation used to apply this treatment

Output Messages:

One of the following messages appears :
1. 'A TRT-GLOBAL with the same identifier exists already'
2. 'Access path between TRT-GLOBAL and EXPERIMENT not created'
3. 'Access path between TRT-GLOBAL and TREATMENT not created'
4. 'This TRT-GLOBAL is not created'
5. 'A new TRT-GLOBAL is created'

Expected interaction and change in the data base : The function compares trt.name in the data base. If such a name exists, it creates only this unique TRT-GLOBAL relation and connects it with the desired EXPERIMENT and TREATMENT. If such a treatment does not exist, it first of all creates an occurrence of the TREATMENT and then introduces the new TRT-GLOBAL relation verifying its constraints of uniqueness. A change in the data base is confirmed by the output message 5. This change means that an occurrence of entity TRT-GLOBAL is added in the data base and an occurrence of the entity TREATMENT if necessary. The other output messages confirm that there is no change in the data base.

5.1.4. SUBFUNCTION: **Particular Observation creation**

Aim: To introduce the information concerning an observation made from one or more units in particular for an experiment conducted.

Input Parameters:

exp.num : experiment number
o.obj: observation object
u.num : unit number
op.date : date of a particular observation made
op.time : time of a particular observation made
op.qnty : quantity of a particular observation made
op.meas : measure used for the quantity
op.level : level of observation
op.method : method used to make this observation

Output Messages:

One of the following messages appears :
1. 'A OBS-PARTIC with the same identifier exists already'
2. 'Access path between OBS-PARTIC and EXPERIMENT not created'
3. 'Access path between OBS-PARTIC and UNIT not created'
4. 'Access path between OBS-PARTIC and OBSERVATION not created'
5. 'This OBS-PARTIC is not created'
6. 'A new OBS-PARTIC is created'

Expected interaction and change in the data base : The function compares obs.obj in the data base. If such an object exists, it creates only this unique OBS-PARTIC relation and connects it with the desired EXPERIMENT, UNIT and OBSERVATION. If such an observation does not exist, it first of all creates an occurrence of the OBSERVATION and then introduces the new OBS-PARTIC relation verifying its constraints of uniqueness. A change in the data base is confirmed by the output message 6. This change means that an occurrence of entity OBS-PARTIC is added in the data base and an occurrence of the entity OBSERVATION if necessary. The other output messages confirm that there is no change in the data base.

5.1.5. SUBFUNCTION: **Global Observation creation**

Aim: To introduce the information concerning an observation made from an experimental site globally for an experiment conducted.

Input Parameters:

exp.num : experiment number
o.obj : observation object
og.date : date of a global observation made
og.time : time of a global observation made
og.qnty : quantity of a global observàtion made
og.meas : measure used for the quantity
og.method : method used to make this observation

Output Messages:

One of the following messages appears :
1. 'A OBS-GLOBAL with the same identifier exists already'
2. 'Access path between OBS-GLOBAL and EXPERIMENT not created'
3.'Access path between OBS-GLOBAL and OBSERVATION not created'
4. 'This OBS-GLOBAL is not created'
5. 'A new OBS-GLOBAL is created'

Expected interaction and change in the data base : The function compares obs.obj in the data base. If such an object exists, it creates only this unique OBS-GLOBAL relation and connects it with the desired EXPERIMENT and OBSERVATION. If such an observation does not exist, it first of all creates an occurrence of the OBSERVATION and then introduces the new OBS-GLOBAL relation verifying its constraints of uniqueness. A change in the data base is confirmed by the output message 5. This change means that an occurrence of entity OBS-GLOBAL is added in the data base and an occurrence of the entity OBSERVATION if necessary. The other output messages confirm that there is no change in the data base.

5.1.6. SUBFUNCTION: **Particular Problem creation**

Aim: To introduce the information concerning a problem encountered in one or more units in particular for an experiment conducted.

Input Parameters:

exp.num : experiment number
p.name: problem name
u.num : unit number
pp.date : date of a particular problem encountered
pp.time : time of a particular problem encountered

Output Messages:

One of the following messages appears :

1. 'A PR-PARTIC with the same identifier exists already'
2. 'Access path between PR-PARTIC and EXPERIMENT not created'
3. 'Access path between PR-PARTIC and UNIT not created'
4. 'Access path between PR-PARTIC and PROBLEM not created'
5. 'This PR-PARTIC is not created'
6. 'A new PR-PARTIC is created'

Expected interaction and change in the data base : The function compares p.name in the data base. If such a name exists, it creates only this unique PR-PARTIC relation and connects it with the desired EXPERIMENT, UNIT and PROBLEM. If such a problem does not exist, it first of all creates an occurrence of the PROBLEM and then introduces the new PR-PARTIC relation verifying its constraints of uniqueness. A change in the data base is confirmed by the output message 6. This change means that an occurrence of entity PR-PARTIC is added in the data base and an occurrence of the entity PROBLEM if necessary. The other output messages confirm that there is no change in the data base.

5.1.7. SUBFUNCTION: **Global Problem creation**

Aim: To introduce the information concerning a problem encountered globally in an experiment.

Input Parameters:

exp.num : experiment number
p.name: problem name
pg.date : date of a global problem encountered
pg.time : time of a global problem encountered

Output Messages:

One of the following messages appears :

1. 'A PR-GLOBAL with the same identifier exists already'
2. 'Access path between PR-GLOBAL and EXPERIMENT not created'
3. 'Access path between PR-GLOBAL and PROBLEM not created'
4. 'This PR-GLOBAL is not created'
5. 'A new PR-GLOBAL is created'

Expected interaction and change in the data base : The function compares p.name. in the data base. If such a name exists, it creates only this unique PR-GLOBAL relation and connects it with the desired EXPERIMENT and PROBLEM. If such a problem does not exist, it first of all creates an occurrence of the PROBLEM and then introduces the new PR-GLOBAL relation verifying its constraints of uniqueness. A change in the data base is confirmed by the output message 5. This change means that an occurrence of entity PR-GLOBAL is added in the data base and an occurrence of the entity PROBLEM if necessary. The other output messages confirm that there is no change in the data base.

5.2. FUNCTION: **INFORMATION DELETION**

Here we present only the subfunctions block deletion, particular treatment deletion, global treatment deletion, particular observation deletion, global observation deletion, particular problem deletion and global problem deletion. The other subfunctions, namely, station deletion, site deletion, climate deletion, experiment deletion, design deletion, observation deletion, treatment deletion and problem deletion will be developed in the future.

5.2.1. SUBFUNCTION: **Block deletion**

Aim: To remove a block and all units contained in this block from the data base, either in case of wrong entry or if it is judged irrelevant for its existence in the data base after a certain period of time.

Input Parameters:

exp.num : experiment number
b.num : block number

Output Messages:

One of the following messages appears on the screen

1. 'Block is occupied by another experiment, can't be deleted'
2. 'Connected to TRT-PARTIC, can't be deleted'
3. 'Connected to PR-PARTIC, can't be deleted'
4. 'Connected to OBS-PARTIC, can't be deleted'
5. 'Block is not deleted from the data base'
6. 'Block is deleted from the data base'

Operation and change in the data base due to this subfunction : The function identifies the desired block in the data base by the b.num and finds out whether any other experiment other than the one identified by exp.num has used this block. If it is used by another experiment, then the message 5 appears. If it is not used and if no one of its units is connected with the relation TRT-PARTIC or PR-PARTIC or OBS-PARTIC, then this block is deleted from the data base and the message 6 appears. Otherwise one of the other messages above appears.

5.2.2. SUBFUNCTION: **Particular Treatment deletion**

Aim: To remove all information concerning a particular treatment relation TRT-PARTIC from the data base.

Input Parameters:

exp.num : experiment number
u.num : unit number
trt.name : treatment name
tp.date : date of a particular treatment
tp.time : time of a particular treatment

Output Messages:

One of the following messages appears on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. 'UNIT for the given identifier is not found'
3. 'TREATMENT for the given identifier is not found'
4. 'TRT-PARTIC for the given date and time is not found'
5. 'TRT-PARTIC is not deleted from the data base'
6. 'TRT-PARTIC is deleted from the data base'

Operation and change in the data base due to this subfunction : Identification of the TRT-PARTIC relation is done in two levels; first by the identifiers of EXPERIMENT, TREATMENT and UNIT, then by the date and time within this group. Once the desired TRT-PARTIC is identified and then is deleted, the message 6 appears on the screen to show that this compound relation TRT-PARTIC is removed from the data base. Otherwise no change takes place in the data base and one of the other messages above appears.

5.2.3. SUBFUNCTION: **Global Treatment deletion**

Aim: To remove all information concerning a global treatment relation TRT-GLOBAL from the data base.

Input Parameters:

exp.num : experiment number
trt.name : treatment name
tg.date : date of a global treatment
tg.time : time of a global treatment

Output Messages:

One of the following messages appears on the screen :
1. 'EXPERIMENT for the given identifier is not found'
2. 'TREATMENT for the given identifier is not found'
3. 'TRT-GLOBAL for the given date and time is not found'
4. 'TRT-GLOBAL is not deleted from the data base'
5. 'TRT-GLOBAL is deleted from the data base'

Operation and change in the data base due to this subfunction: Identification of the TRT-GLOBAL relation is done in two levels; first by the identifiers of EXPERIMENT and TREATMENT ; then by the date and time within this group. Once the desired TRT-GLOBAL is identified and then deleted, the message 5 appears on the screen to show that this compound relation TRT-GLOBAL is removed from the data base. Otherwise no change takes place in the data base and one of the other messages above appears.

5.2.4. SUBFUNCTION: **Particular Observation deletion**

Aim: To remove all information concerning a particular observation relation OBS-PARTIC from the data base.

Input Parameters:

exp.num : experiment number
u.num : unit number
o.obj : observation object
op.date : date of a particular observation
op.time : time of a particular observation

Output Messages:

One of the following messages appears on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. 'UNIT for the given identifier is not found'
3. 'OBSERVATION for the given identifier is not found'
4. 'OBS-PARTIC for the given date and time is not found'
5. 'OBS-PARTIC is not deleted from the data base'
6. 'OBS-PARTIC is deleted from the data base'

Operation and change in the data base due to this subfunction: Identification of the OBS-PARTIC relation is done in two levels; first by the identifiers of EXPERIMENT, OBSERVATION and UNIT, then by the date and time within this group. Once the desired OBS-PARTIC is identified and then is deleted, the message 6 appears on the screen to show that this compound relation OBS-PARTIC is removed from the data base. Otherwise no change takes place in the data base and one of the other messages above appears.

5.2.5. SUBFUNCTION: **Global Observation deletion**

Aim: To remove all information concerning a global observation relation OBS-GLOBAL from the data base.

Input Parameters:

exp.num : experiment number
o.obj : observation object
og.date : date of a global observation
og.time : time of a global observation

Output Messages:

One of the following messages appears on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. 'OBSERVATION for the given identifier is not found'
3. 'OBS-GLOBAL for the given date and time is not found'
4. 'OBS-GLOBAL is not deleted from the data base'
5. 'OBS-GLOBAL is deleted from the data base'

Operation and change in the data base due to this subfunction: Identification of the OBS-GLOBAL relation is done in two levels; first by the identifiers of EXPERIMENT and OBSERVATION, then by the date and time within this group. Once the desired OBS-GLOBAL is identified and then is deleted, the message 5 appears on the screen to show that this compound relation OBS-GLOBAL is removed from the data base. Otherwise no change takes place in the data base and one of the other messages above appears.

5.2.6. SUBFUNCTION: **Particular Problem deletion**

Aim: To remove all information concerning a particular problem relation PR-PARTIC from the data base.

Input Parameters:

exp.num : experiment number
u.num : unit number
p.name : problem name
pp.date : date of a particular problem
pp.time : time of a particular problem

Output Messages:

One of the following messages appears on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. 'UNIT for the given identifier is not found'
3. 'PROBLEM for the given identifier is not found'
4. 'PR-PARTIC for the given date and time is not found'
5. 'PR-PARTIC is not deleted from the data base'
6. 'PR-PARTIC is deleted from the data base'

Operation and change in the data base due to this subfunction: Identification of the PR-PARTIC relation is done in two levels; first by the identifiers of EXPERIMENT, PROBLEM and UNIT, then by the date and time within this group. Once the desired PR-PARTIC is identified and then is deleted, the message 6 appears on the screen to show that this compound relation PR-PARTIC is removed from the data base. Otherwise no change takes place in the data base and one of the other messages above appears.

5.2.7. SUBFUNCTION: **Global Problem deletion**

Aim: To remove all information concerning a global problem relation PR-GLOBAL from the data base.

Input Parameters:

exp.num : experiment number
p. name : problem name
pg. date : date of a global problem
pg. time : time of a global problem

Output Messages:

One of the following messages appears on the screen :
1. 'EXPERIMENT for the given identifier is not found'
2. 'PROBLEM for the given identifier is not found'
3. 'PR-GLOBAL for the given date and time is not found'
4. 'PR-GLOBAL is not deleted from the data base'
5. 'PR-GLOBAL is deleted from the data base'

Operation and change in the data base due to this subfunction: Identification of the PR-GLOBAL relation is done in two levels; first by the identifiers of EXPERIMENT and PROBLEM, then by the date and time within this group. Once the desired PR-GLOBAL is identified and then is deleted, the message 5 appears on the screen to show that this compound relation PR-GLOBAL is removed from the data base. Otherwise no change takes place in the data base and one of the other messages above appears.

5.3. FUNCTION: **INFORMATION MODIFICATION**

We expose here only the subfunctions experiment modification, particular treatment modification, global treatment modification, particular observation modification, global observation modification, particular problem modification, global problem modification, unit modification and block modification. The remaining subfunctions, namely, station modification, climate modification, site modification, design modification, treatment modification, observation modification and problem modification will be developed in the future.

5.3.1. SUBFUNCTION: **Experiment modification**

Aim: To introduce certain changes in the values of the attributes of an entity EXPERIMENT in the data base.

Input Parameters:

exp.num : experiment number
exp.stdate : experiment starting date
exp.endate : experiment ending date
exp.obj : experiment objective

Output Messages:

One of the following messages appears on the screen :
1. 'EXPERIMENT for the given identifier is not found'
2. 'EXPERIMENT is not modified in the data base'
3. 'EXPERIMENT has been modified in the data base'

Possible action and change in the data base : The function identifies the EXPERIMENT to be modified by its identifier exp.num. Since the identifier of the EXPERIMENT is created respecting its uniqueness by the program during the design layout, only other attribute values may have to be changed. If the values of the attributes are updated, the message number 3 appears on the screen indicating the change in the data base. Otherwise no change takes place in the data base and either the message 1 or 2 above appears on the screen.

5.3.2. SUBFUNCTION: **Particular Treatment modification**

Aim: To introduce certain changes in the values of the attributes of a particular treatment relation TRT-PARTIC in the data base.

Input Parameters:

exp.num : experiment number
trt.name : treatment name
u.num : unit number
tp.odate : old date of the particular treatment
tp.otime : old time of the particular treatment
tp.ndate : new date of the particular treatment
tp.ntime : new time of the particular treatment

tp.qnty : quantity of a particular treatment applied

tp.meas : measure used for the quantity

tp.level : if this treatment is applied as one of the factors, then the level is marked

tp.type : type of the treatment applied

tp.opertype : type of operation used to apply this treatment

Note : For the last seven parameters, the new values are entered if they are to be modified and the original values are entered if they don't need modification.

Output Messages:

One of the following messages is displayed on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. 'TREATMENT for the given identifier is not found'
3. 'UNIT for the given identifier is not found'
4. 'A TRT-PARTIC with the same identifier exists already'
5. 'TRT-PARTIC is not modified in the data base'
6. 'TRT-PARTIC has been modified in the data base'
7. 'TRT-PARTIC for the given identifier is not found'

Possible action and change in the data base : If at least one of the new values among date and time is different from the values of the original date and time, then the function searches the data base to find out whether a relation TRT-PARTIC exists with new date and time. If it exists, the message 4 is displayed on the screen; otherwise if the new values of date and time are the same as the original or if for the new values of date and time there does not exist a TRT-PARTIC in the data base, then the relation TRT-PARTIC for the original values of date and time is searched in the data base and modified while it is found. In this way, the uniqueness of the identifier is safeguarded. If it is modified, then the message 6 appears on the screen. If the relation is not found for the original values of date and time, the message 7 appears. Identification of this relation is first done through EXPERIMENT, UNIT and TREATMENT by exp.num, u.num and trt.name respectively and then by date and time. Except for the message 6 there is no change in the data base.

5.3.3. SUBFUNCTION: **Global Treatment modification**

Aim: To introduce certain changes in the values of the attributes of a global treatment relation TRT-GLOBAL in the data base.

Input Parameters:

exp.num : experiment number
trt.name : treatment name
tg.odate : old date of the global treatment
tg.otime : old time of the global treatment
tg.ndate : new date of the global treatment
tg.ntime : new time of the global treatment
tg.qnty : quantity of a global treatment applied
tg.meas : measure used for the quantity
tg.opertype : type of operation used to apply this treatment

Note : For the last five parameters the new values are entered if they are to be modified and the original values are entered if they don't need modification.

Output Messages:

One of the following messages is shown on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. 'TREATMENT for the given identifier is not found'
3. 'A TRT-GLOBAL with the same identifier exists already'
4. 'TRT-GLOBAL is not modified in the data base'
5. 'TRT-GLOBAL has been modified in the data base'
6. 'TRT-GLOBAL for the given identifier is not found'

Possible action and change in the data base : If at least one of the new values among date and time is different from the values of the original date and time, then the function searches the data base to find out whether a relation TRT-GLOBAL exists with new date and time. If it exists, the message 3 is displayed on the screen; otherwise if the new values of date and time are the same as the original or if for the new values of date and time there does not exist a TRT-GLOBAL in the data base, then the relation TRT-GLOBAL for the original values of date and time is searched in the data base and modified while it is found. In this way the uniqueness of the identifier of the relation TRT-GLOBAL is safeguarded. If it is modified, then the message 5 appears on the screen. If the relation is not found for the original values of date and time, the message 6 appears. Identification of this relation is first done through EXPERIMENT and TREATMENT by exp.num and trt.name respectively and then by date and time. Except for the message 5 there is no change in the data base.

5.3.4. SUBFUNCTION: **Particular Observation modification**

Aim: To introduce certain changes in the values of the attributes of a particular observation relation OBS-PARTIC in the data base.

Input Parameters:

exp.num : experiment number
o.obj : observation object
u.num : unit number
op.odate : old date of the particular observation
op.otime : old time of the particular observation
op.ndate : new date of the particular observation
op.ntime : new time of the particular observation
op.qnty : quantity of a particular observation made
op.meas : measure used for the quantity
op.level : level of a particular observation
op.method : method used for a particular observation

Note : For the last six parameters, the new values are entered if they are to be modified and the original values are entered if they don't need modification.

Output Messages:

One of the following messages appears on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. 'OBSERVATION for the given identifier is not found'
3. 'UNIT for the given identifier is not found'
4. 'An OBS-PARTIC with the same identifier exists already'
5. 'OBS-PARTIC is not modified in the data base'
6. 'OBS-PARTIC has been modified in the data base'
7. 'OBS-PARTIC for the given identifier is not found'

Possible action and change in the data base : If at least one of the new values among date and time is different from the values of the original date and time, then the function searches the data base to find out whether a relation OBS-PARTIC exists with new date and time. If it exists, the message 4 is displayed on the screen; otherwise, if the new values of date and time are the same as the original or if for the new values of date and time there does not exist an OBS-PARTIC in the data base, then the relation OBS-PARTIC for the original values of date and time is searched in the data base and modified while it is found. In this way, the uniqueness of the identifier is safeguarded. If

it is modified, then the message 6 appears on the screen. If the relation is not found for the original values of date and time, the message 7 appears. Identification of this relation is first done through EXPERIMENT, UNIT and OBSERVATION by exp.num, u.num and o.obj respectively and then by date and time. Except for the message 6, there is no change in the data base.

5.3.5. SUBFUNCTION: **Global Observation modification**

Aim: To introduce certain changes in the values of the attributes of a global observation relation OBS-GLOBAL in the data base.

Input Parameters:

exp.num : experiment number
o.obj : observation object
og.odate : old date of the global observation
og.otime : old time of the global observation
og.ndate : new date of the global observation
og.ntime : new time of the global observation
og.qnty : quantity of a global observation made
og.meas : measure used for the quantity
og.method : method used for a global observation

Note : For the last five parameters, the new values are entered if they are to be modified and the original values are entered if they don't need modification.

Output Messages:

One of the following messages is displayed on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. 'OBSERVATION for the given identifier is not found'
3. 'An OBS-GLOBAL with the same identifier exists already'
4. 'OBS-GLOBAL is not modified in the data base'
5. 'OBS-GLOBAL has been modified in the data base'
6. 'OBS-GLOBAL for the given identifier is not found'

Possible action and change in the data base : If at least one of the new values among date and time is different from the values of the original date and time, then the function searches the data base to find out whether a relation OBS-GLOBAL exists with new date and time. If it exists, the message 3 is displayed on the screen; otherwise, if the new values of date and time are the same as the original or if for the new values of date and

time there does not exist an OBS-GLOBAL in the data base, then the relation OBS-GLOBAL for the original values of date and time is searched in the data base and modified while it is found. In this way, the uniqueness of the identifier is safeguarded. If it is modified, then the message 5 appears on the screen. If the relation is not found for the original values of date and time, the message 6 appears. The identification of this relation is first done through EXPERIMENT and OBSERVATION by exp.num and o.obj respectively and then by date and time. Except for the message 5 there is no change in the data base.

5.3.6. SUBFUNCTION: **Particular Problem modification**

Aim: To introduce certain changes in the values of the attributes of a particular problem relation PR-PARTIC in the data base.

Input Parameters:

exp.num : experiment number
p.name: problem name
u.num : unit number
pp.odate : old date of the particular problem
pp.otime : old time of the particular problem
pp.ndate : new date of the particular problem
pp.ntime : new time of the particular problem

Output Messages:

One of the following messages is displayed on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. 'PROBLEM for the given identifier is not found'
3. 'UNIT for the given identifier is not found'
4. 'A PR-PARTIC with the same identifier exists already'
5. 'PR-PARTIC is not modified in the data base'
6. 'PR-PARTIC has been modified in the data base'
7. 'PR-PARTIC for the given identifier is not found'

Possible action and change in the data base : The function searches the data base to find out whether a relation PR-PARTIC exists with new date and time. If it exists, the message 4 is displayed on the screen; otherwise, the relation PR-PARTIC for the original values of date and time is searched in the data base and modified while it is found. In this way, the uniqueness of the identifier of the relation PR-PARTIC is safeguarded. If it is

modified, then the message 6 appears on the screen. If the relation is not found for the original values of date and time, message 7 appears. Except for the message 6, there is no change in the data base. Identification of this relation is first done through EXPERIMENT, UNIT and PROBLEM by exp.num, u.num and p.name respectively and then by date and time.

5.3.7. SUBFUNCTION: **Global Problem modification**

Aim: To introduce certain changes in the values of the attributes of a global problem relation PR-GLOBAL in the data base.

Input Parameters:

exp.num : experiment number
p.name : problem name
pg.odate : old date of the global problem
pg.otime : old time of the global problem
pg.ndate : new date of the global problem
pg.ntime : new time of the global problem

Output Messages:

One of the following messages is displayed on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. 'PROBLEM for the given identifier is not found'
3. 'A PR-GLOBAL with the same identifier exists already'
4. 'PR-GLOBAL is not modified in the data base'
5. 'PR-GLOBAL has been modified in the data base'
6. 'PR-GLOBAL for the given identifier is not found'

Possible action and change in the data base : The function searches the data base to find out whether a relation PR-GLOBAL exists with new date and time. If it exists, the message 3 is displayed on the screen; otherwise, the relation PR-GLOBAL for the original values of date and time is searched in the data base and modified while it is found. In this way, the uniqueness of the identifier of the relation PR-GLOBAL is safeguarded. If it is modified, then the message 5 appears on the screen. If the relation is not found for the original values of date and time, the message 6 appears. Except for the message 5, there is no change in the data base. The identification of this relation is first done through EXPERIMENT and PROBLEM by exp.num, and p.name respectively and then by date and time.

5.3.8. SUBFUNCTION: **Unit modification**

Aim: To introduce certain changes in the values of the attributes of an entity UNIT in the data base.

Input Parameters:

u.num : unit number
u.rnum : unit row number
u.cnum : unit column number

Output Messages:

One of the following messages appears on the screen :
1. 'UNIT for the given identifier is not found'
2. 'UNIT is not modified in the data base'
3. 'UNIT has been modified in the data base'

Possible action and change in the data base : The function identifies the UNIT to be modified by its identifier u.num. Since the identifier of the UNIT is created respecting its uniqueness by the program during the design layout, only other attribute values may have to be changed. If the values of the attributes are updated, the message number 3 appears on the screen, indicating the change in the data base. Otherwise, no change takes place in the data base and either message 1 or 2 above appears on the screen.

5.3.9. SUBFUNCTION: **Block modification**

Aim: To introduce certain changes in the values of the attributes of an entity BLOCK in the data base.

Input Parameters:

b.num : block number
b.rnum : block row number
b.cnum : block column number
b.type : block type

Output Messages:

One of the following messages appears on the screen :

1. 'BLOCK for the given identifier is not found'
2. 'BLOCK is not modified in the data base'
3. 'BLOCK has been modified in the data base'

Possible action and change in the data base : The function identifies the BLOCK to be modified by its identifier b.num. Since the identifier of the BLOCK is created respecting its uniqueness by the program during the design layout, only other attribute values may have to be changed. If the values of the attributes are updated, the message number 3 appears on the screen indicating the change in the data base. Otherwise, no change takes place in the data base and either the message 1 or 2 above appears on the screen.

CHAPTER 6

INFORMATION RETRIEVAL MODULE

We describe in this module only the functions site occupation information, experiment information, particular treatment information, global treatment information, particular observation information, global observation information, particular problem information and global problem information. Some more information are retreived according to the needs of the display layout function in the Statistical Design Layout Module and in the Statistical Analysis Module. Other information retrieval functions can be developed depending on the future need of the application.

6.1. FUNCTION: SITE OCCUPATION INFORMATION

Aim: To consult the data base for informations concerning the experimental site used by experiments as well as reserved for experiments in the future in a research station.

Input Parameter:

st.name : station name

Output Messages:

One of the two following messages appears on the screen :

1. 'STATION for the given identifier is not found'
2. For each site used or reserved for experiments, the following information is provided :

    s.num : site number

    s.type : site type

    s.size : site size

    a group of experiments which has occupied this site, each providing the following information :

        exp.num : experiment number

        exp.stdate : experiment starting date

        exp.endate : experiment ending date

Operation: A station is identified by its name (st.name). If it is not found, the message 1 is displayed on the screen. If at least one site in the station is used or reserved for an experiment, then the message 2 appears.

6.2. FUNCTION: **EXPERIMENT INFORMATION**

Aim: To retrieve information from the data base concerning an EXPERIMENT conducted in a research station.

Input Parameters:

exp.num : experiment number

Output Messages:

One of the following messages is displayed on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. exp.stdate : experiment starting date
   exp.endate : experiment ending date
   exp.obj : experiment objective

Operation: An EXPERIMENT is identified by its number exp.num. If it is found, the message 1 above appears on the screen; otherwise, the message 2 is displayed.

6.3. FUNCTION: **PARTICULAR TREATMENT INFORMATION**

Aim: To consult the data base regarding the information of one or more particular treatments of a specific experiment conducted in a research station.

Input Parameters:

exp.num : experiment number
u.num : unit number
trt.name : treatment name

Output Messages:

One of the following messages is displayed on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. 'TREATMENT for the given identifier is not found'
3. 'UNIT for the given identifier is not found'
4. tp.qnty : quantity of the particular treatment
   tp.meas : measure of the particular treatment
   tp.level : level of the particular treatment
   tp.type : type of the particular treatment

tp.opertype : operation type of the particular treatment
tp.date : date of application of the particular treatment

Operation: If only the proper EXPERIMENT and for each relation TRT-PARTIC its TREATMENT and UNIT are identified successfully, the function displays the message 4 above; otherwise, it displays one of the other messages above. The message 4 will appear as many times as there are existing TRT-PARTIC relations in the data base for the given input parameters.

6.4. FUNCTION: **GLOBAL TREATMENT INFORMATION**

Aim: To consult the data base regarding the information of all the global treatments of a specific experiment conducted in a research station.

Input Parameters:

exp.num : experiment number

Output Messages:

One of the following messages is displayed on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. 'TREATMENT for the given identifier is not found'
3. tg.qnty : quantity of the global treatment
   tg.meas : measure of the global treatment
   tg.opertype : operation type of the global treatment
   tg.date : date of application of the global treatment
   trt.name : treatment name

Operation: If only the proper EXPERIMENT and for each relation TRT-GLOBAL its TREATMENT is identified successfully, the function displays the message 3 above; otherwise, it displays one of the other messages above. The message 3 will appear as many times as there are existing TRT-GLOBAL relations in the data base for the given experiment number (exp.num).

6.5. FUNCTION: **PARTICULAR OBSERVATION INFORMATION**

Aim: To consult the data base regarding the information of one or more particular observations of a specific experiment conducted in a research station.

Input Parameters:

exp.num : experiment number

u.num : unit number

o.obj : object of observation

Output Messages:

One of the following messages is displayed on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. 'OBSERVATION for the given identifier is not found'
3. 'UNIT for the given identifier is not found'
4. op.qnty : quantity of the particular observation
   op.meas : measure of the particular observation
   op.level : level of the particular observation
   op.method : method employed for the particular observation
   op.date : date of application of the particular observation

Operation: If only the proper EXPERIMENT and for each relation OBS-PARTIC its OBSERVATION and UNIT are identified successfully, the function displays the message 4 above; otherwise, it displays one of the other messages above. The message 4 will appear as many times as there are existing OBS-PARTIC relations in the data base for the given input parameters.

6.6. FUNCTION: GLOBAL OBSERVATION INFORMATION

Aim: To consult the data base regarding the information of all the global observations of a specific experiment conducted in a research station.

Input Parameters:

exp.num : experiment number

Output Messages:

One of the following messages is displayed on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. 'OBSERVATION for the given identifier is not found'
3. og.qnty : quantity of the global observation
   og.meas : measure of the global observation
   og.method : method employed for the global observation
   og.date : date of application of the global observation
   o.obj : object of observation

Operation: If only the proper EXPERIMENT and for each relation OBS-GLOBAL its OBSERVATION is identified successfully, the function displays the message 3 above; otherwise, it displays one of the other messages above. The message 3 will appear as many times as there are existing OBS-GLOBAL relations in the data base for the given experiment number (exp.num).

6.7. FUNCTION: **PARTICULAR PROBLEM INFORMATION**

Aim: To consult the data base regarding the information of all the particular problems of a specific experiment conducted in a research station.

Input Parameters:

exp.num : experiment number

Output Messages:

One of the following messages is displayed on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. 'PROBLEM for the given identifier is not found'
3. 'UNIT for the given identifier is not found'
4. pp.date : date of the particular problem encountered
   pp.time : time of the particular problem occurring
   u.num : unit number
   p.name : problem name

Operation: If only the proper EXPERIMENT and for each relation PR-PARTIC its PROBLEM and UNIT are identified successfully, the function displays the message 4 above; otherwise, it displays one of the other messages above. The message 4 will appear as many times as there are existing PR-PARTIC relations in the data base for the given experiment number (exp.num).

6.8. FUNCTION: **GLOBAL PROBLEM INFORMATION**

Aim: To consult the data base regarding the information of all the global problems of a specific experiment conducted in a research station.

Input Parameters:

exp.num : experiment number

Output Messages:

One of the following messages is displayed on the screen :

1. 'EXPERIMENT for the given identifier is not found'
2. 'PROBLEM for the given identifier is not found'
3. pg.date : date of the global problem encountered
   pg.time : time of the global problem occurring
   p.name : problem name

Operation: If only the proper EXPERIMENT and for each relation PR-GLOBAL its PROBLEM is identified successfully, the function displays the message 3 above; otherwise, it displays one of the other messages above. The message 3 will appear as many times as there are existing PR-GLOBAL relations in the data base for the given experiment number (exp.num).

CHAPTER 7

STATISTICAL DESIGN LAYOUT MODULE

In this module, the first part is devoted to designing a manual layout for the experiment implemented in the past for Randomized Complete Block Design (R.B.D.), R.B.D. layout for Mixed Factorial Design and Split-Plot Design. In the second part, we describe the function for designing an automatic layout for the new experiment to be conducted for the above mentioned three designs. Finally, in the third part, we present a description of the function Display Layout. This function will be the same for all the designs cited above.

7.1. FUNCTION: **DESIGNING MANUAL LAYOUT**

7.1.1. SUBFUNCTION: **Randomized Block Design Layout**

Aim: To introduce the information regarding an experiment, its design, the site(s) it occupies, its blocks with their units and the *a priori* label allotment of particular treatments in each and every unit.

Input Parameters:

1. exp.stdate : experiment starting date
   exp.endate : experiment ending date
   exp.obj : objective of the experiment
   d.name : design name
   d.desc : design description
   total number of blocks

2. *For each block*
   s.num : site number it occupies
   b.rnum : block row number in the layout
   b.cnum : block column number in the layout
   b.type : block type
   total number of units for this block
   For each unit
   u.rnum : unit row number
   u.cnum : unit column number

3. The values of the following parameters are the same for all the particular treatment relations :

   tp.date : date of particular treatment allotted

   tp.time : time of particular treatment allotted

   tp.type : type of each particular treatment

   tp.opertype : type of operation used for the particular treatment

   (*a priori* allotment type)

   maximum number of treatments for the treatment combination

4. The values of the following parameters are entered for the table of treatment combinations :

   trt.name : name of treatment(s) for each treatment combination

   tp.qnty : quantity of each particular treatment to be applied

   tp.meas : measure used for the quantity

   tp.level : level of each particular treatment (rank of treatment combination) for each unit starting from the first, the rank number

Note : A rank is a serial number from 1 to n, where n is the total number of treatment combinations, in order to refer to the treatment combinations in a particular order. A treatment combination may contain one or more particular treatments. If a treatment combination contains more than one particular treatment, then each particular treatment in that combination will receive as "level" number the same rank number as that of the treatment combination (this arrangement is meant only for R.B.D. designs.)

Output Message:

1. 'A new EXPERIMENT is created in the data base'
2. 'This EXPERIMENT is not created in the data base'
3. 'Please note your experiment number and the first unit identification number in your diary'
4. exp.num : experiment number; first unit identification number
5. 'A new DESIGN has been created in the data base'
6. 'This DESIGN is not created in the data base'
7. 'Access path between DESIGN and EXPERIMENT is not inserted'
8. 'SITE for the given identifier is not found'
9. 'This OCCUPY is not created in the data base'
10. 'Access path between OCCUPY and SITE is not inserted'

11. 'Access path between BLOCK and OCCUPY is not inserted'
12. 'Access path between EXPERIMENT and OCCUPY is not inserted'
13. 'Access path between BLOCK and UNIT is not inserted'
14. 'This UNIT has not been created in the data base'
15. 'A new BLOCK is created in the data base'
16. 'A new TREATMENT is created in the data base'
17. 'A TREATMENT with same identifier exists already'
18. 'This TREATMENT is not created in the data base'
19. 'A new TRT-PARTIC is created in the data base'
20. 'This TRT-PARTIC is not created in the data base'
21. 'A TRT-PARTIC with same identifier exists already'
22. 'Access path between TREATMENT and TRT-PARTIC is not inserted'
23. 'Access path between UNIT and TRT-PARTIC is not inserted'
24. 'Access path between EXPERIMENT and TRT-PARTIC is not inserted'
25. 'Access path between DESIGN and EXPERIMENT is inserted'

Operation and Change in the data base :

(1) This function creates the entity EXPERIMENT. If the EXPERIMENT has been successfully created in the data base, the messages 1, 3 and 4 appear on the screen and the function proceeds to create a DESIGN entity; otherwise, the message 2 appears and once again the creation commences with the necessary corrections.

(2) The function creates a DESIGN entity if it does not exist already in the data base for the given d.name and if it does exist, it merely inserts an access path between the DESIGN and the EXPERIMENT just created and the message 25 appears. If a new DESIGN is created, the message 5 is displayed and the function proceeds to the creation of the entity BLOCK, its UNITs and its relation OCCUPY with SITE(s) and EXPERIMENT; otherwise, either the message 6 and 7 appears and the process restarts after the necessary corrections.

(3) If the BLOCK is successfully created, the message 15 is shown on the screen and the operation continues to create the next BLOCK and so on until all the BLOCKs are created; otherwise, one of the messages from 8 to 14 appears and the creation operation for that particular BLOCK has to be started again.

(4) Once all the BLOCKs have been created, the user proceeds to introduce the information concerning the particular treatment label (*a priori*) allotment (third part of the

input parameters above). The function forms the treatment combination table in the order in which the user wants. Once it receives a rank number from the user, it identifies the treatment combination associated with this number from the table of treatment combinations and creates the TRT-PARTIC relation for the first unit in the first block. Note that this function first creates the TREATMENT entity if necessary and then creates the TRT-PARTIC relation. If one of the messages, namely, 16 and 17 appears, then the operation proceeds to the creation of TRT-PARTIC relation. The message 19 indicates the success of the operation for each particular treatment creation; otherwise, one of the messages from 20 to 24 appears and the function repeats the procedure with the necessary corrections, if any until it succeeds in creating the occurrence of this relation in the data base.

As the user gives the rank number of the treatment combination for each unit, the function creates the particular treatment combination. This procedure is repeated until the creation of all the TRT-PARTIC entities for each unit, from block to block, has been completed.

Note: This function is meant only for an R.B.D. experiment for which a layout has already been implemented in the past.

7.1.2. SUBFUNCTION: **R.B.D. Layout For Mixed Factorial Design**

Aim: It is the same as in the subfunction *designing manual layout for R.B.D.* (cf 7.1.1.)

Input Parameters:

Up to the second part they are the same as in subfunction *designing manual layout for R.B.D.* (cf 7.1.1.). Instead of the third and the fourth parts we introduce the following parameters :

1. The values of the following parameters are the same for all the particular treatment relations :

   tp.date : date of particular treatment allotted

   tp.time : time of particular treatment allotted

   tp.type : type of each particular treatment

   tp.opertype : type of operation used for the particular treatment

   (*a priori* allotment type)

2. number of factors (treatments)
   name of each factor
   maximum number of levels for each factor
   tp.qnty for each level of these factors
   tp.meas for each tp.qnty
   for each unit (starting from the first), the combination of levels of the factors allotted (*a priori* allotment)

Output Message:

It is the same as in the subfunction for designing manual layout for R.B.D. (cf 7.1.1.)

Operation and change in the data base : It is the same as in the previous subfunction for *designing manual layout for R.B.D.*, except for the following modification in the fourth part : The user supplies only the treatment combination levels for each unit as they had been allotted in the past. The procedure calculates the rank number and identifies the treatment combination from the table which had been formed from the first five input parameters given in the second part above and creates the particular treatment combination for each unit, block after block.

7.1.3. SUBFUNCTION: **Split-Plot Design Layout**

Aim, input parameters, output messages, operation and changes in the data base are the same as in the previous subfunction for *manual layout for factorial design* except for the following small change : This subfunction associates the first factor with the whole-plot treatment and the second factor with the split-plot treatment for tp.type in each treatment combination and then creates the TRT-PARTIC entities for each unit.

7.2. FUNCTION: **DESIGNING AUTOMATIC LAYOUT**

7.2.1. SUBFUNCTION: **Randomized Block Design Layout**

Aim and output messages are the same as for the first subfunction of the function *designing manual layout.* (cf 7.1.1.)

Input Parameters:

1. exp.stdate : experiment starting date
   exp.endate : experiment ending date
   exp.obj : objective of the experiment
   d.name : design name
   d.desc : design description
   total number of blocks

2. *For each block*
   s.num : site number it occupies
   b.rnum : block row number in the layout
   b.cnum : block column number in the layout
   b.type : block type
   total number of units

3. *For each unit*
   u.rnum : unit row number
   u.cnum : unit column number

4. The values of the following parameters are the same for all the particular treatment relations :
   maximum number of treatments for the treatment combination
   tp.date : date of particular treatment
   tp.time : time of particular treatment
   tp.type : type of each particular treatment
   tp.opertype : type of operation used for the particular treatment
   (label allotment type)

5. The values of the following parameters are entered to form the table of treatment combinations :
   trt.name : name of treatment(s) for each treatment combination
   tp.qnty : quantity of each particular treatment
   tp.meas : measure used for the quantity
   tp.level : "level" (rank) of each particular treatment

Operation and change in the data base :

(1) The procedure for the creation of the entity EXPERIMENT is the same as in the first subfunction of *designing manual layout.* (cf 7.1.1.)

(2) The procedure for the creation of the entity DESIGN is the same as in the first subfunction of *designing manual layout.* (cf 7.1.1.)

(3) The procedure for the creation of the entity BLOCK is the same as in the first subfunction of *designing manual layout.* (cf 7.1.1.)

(4) Once all the BLOCKs have been created, this function creates a table containing the components of a treatment combination. Each record in the table will contain the components trt.name, tp.qnty, tp.meas and tp.level. Here the rank of the treatment combination will be entered against tp.level and the rank or treatment combination number will be identified through the index of the table when randomization procedure generates the treatment number.

(5) Now the function proceeds to the automatic *(a priori)* label allotment of the treatment combinations to each and every unit according to the principle of randomization for Randomized Complete Block Design[1] and chooses one of the treatment combinations from the table formed in t step (4) above.

For R.B.D., randomization is done separetely for each and every block. For a given block, one of its units is chosen at random and one of the treatment combinations chosen at random is allotted to that unit. This procedure continues until all the units in the block are allotted a treatment combination. Note that each unit receives a treatment combination that is different from that of the others in the block. Therefore, there are as many treatment combinations as the number of units in a block. All blocks contain an equal number of units.

The function creates the relations TRT-PARTIC between TREATMENT, EXPERIMENT and UNIT for each randomly chosen unit and particular treatment combination. Note that this function first creates the TREATMENT entity if necessary and then creates the TRT-PARTIC relation. If one of the messages, namely, 16 or 17 appears, then the operation proceeds to the creation of a TRT-PARTIC relation. The message 19 indicates the success of the operation for each particular treatment creation; otherwise, one of the messages from 20 to 24 appears and the function repeats the procedure with the necessary corrections if any until it succeeds in creating the occurrence of this relation in

[1] PANSE, 67

the data base (Please refer to subfunction of *designing manual layout* (cf 7.1.1.) for the above output messages).

7.2.2. SUBFUNCTION: **R.B.D. Layout For Factorial Design**

Aim, input parameters, output messages, operation and change in the data base are the same as in subfunction of *designing automatic R.B.D. layout* (cf 7.2.1.) except for the following omission and addition :

Up to the third part, the input parameters are the same. Then we have the following parameters :

1. The values of the following parameters are the same for all the particular treatment relations :
   - tp.date : date of particular treatment
   - tp.time : time of particular treatment
   - tp.type : type of each particular treatment
   - tp.opertype : type of operation used for the particular treatment
     (label allotment type)

2. number of factors (treatments)
   For each factor :
   - trt.name : name of the factor
   - maximum number of levels for that factor
   - *For each level* :
     - tp.qnty : quantity of that factor level
     - tp.meas : measure used for the quantity

During the **operation**, the procedure generates the treatment combinations and forms the table automatically, unlike the previous subfunction for R.B.D. in which the treatment combinations were entered manually to form the table.

7.2.3. SUBFUNCTION: **Split-Plot Design Layout**

Aim, input parameters, output messages, operation and change in the data base are the same as in the subfunction of *designing automatic R.B.D. layout for factorial design* (cf 7.2.2.), except for the following modifications in the operation :

(1) The arrangement of units in the block has to be in accordance with the number of whole-plot treatments and split-plot treatments. "In field experiments it is natural to make up the split plots within a whole plot in such a way as to sample best the whole plot, i.e., in the direction with the greater trend, and to lay out the whole plots within a replicate in such a way as to sample the replicate best" (KEMP, 73). Therefore, the units will be numbered in such a way that each whole plot is completed one after the other in order. This order will help the automatic random allotment of whole-plot treatment and split-plot treatment. The number of whole plots in a block will be equal to the number of whole-plot treatments and the number of split plots within a whole plot will be equal to the number of split-plot treatments. In our case of split-plot design for factorial experiment, the number of whole plots within a replicate or block will be equal to the total number of levels of the whole-plot treatment and the number of split plots within a whole plot will be equal to the total number of levels of the split-plot treatment.

(2) The randomization procedure is to allot whole-plot treatments at random to the whole plots, and then to allot the split-plot treatments at random within the whole plot (KEMP, 73).

(3) The value of the tp.type for the first factor will be 'whole-plot' and for the second factor, it will be 'split-plot'.

7.3. FUNCTION: **DISPLAY LAYOUT**

Aim: To retrieve the information concerning the block layout and the particular treatment allotted to each and every unit of the block.

Note : The layout will be displayed on the screen graphically as it is physically designed on the experimental site with blocks and units in their proper positions in the layout. In addition, different treatment combinations allotted to the units will be displayed in different colours so as to make it easy for the user to identify the same treatment combinations falling on different blocks and to have an overall view of the random allotments of the different treatment combinations in the experimental site.

Input Parameters:

exp.num : experiment number

tp.opertype : particular treatment operation type

Output Messages:

1. b.num : block numbers
2. treatment combinations
3. name of the design
4. graphical and colour display of the layout
5. 'EXPERIMENT for the given identifier is not found'
6. 'TRT-PARTIC for the given operation type is not found'
7. 'Success! Layout for the given experiment is found'

Operation: If the desired experiment for the display of the layout is not found in the data base, then the message 5 appears on the screen. If the given tp.opertype does not match the one existing in the data base, then the message 6 appears. Otherwise, the function finds out all the blocks together with their units belonging to this experiment and their position in the layout, and the treatment combinations allotted to each unit. After the graphic and colour formation, the output messages 1, 2, 3, 4 and 7 appear on the screen.

Note: This function can be used for any design layout with complete block layout.

CHAPTER 8

STATISTICAL ANALYSIS MODULE

In this chapter, the functions deal with the analysis of Randomized Complete Block Design, Mixed Factorial Design and Split-Plot Design. The bar diagram for blockwise observed particular quantities helps the user to assess the differences present in the experimental material through visual image and to take the necessary steps for better designing of the future experimental layout. The image can be considered as a sort of blockwise fertility gradiant map for agricultural experiments. The notation and the method of presentation for the model and the analysis of variance table used by O.Kempthorne (KEMP, 73) have been adapted to our need here and now.

8.1. FUNCTION: RANDOMIZED BLOCK DESIGN ANALYSIS

Aim: To retrieve the data from the data base concerning the quantities observed on each unit (one observation per unit) for Randomized Block Design experiment (these are the data which had been collected in accordance with an objective of the experiment) and to present a statistical analysis of the data.

Input Parameters:

exp.num : experiment number
tp.opertype : particular treatment operation type (*a priori* allotment type)
o.obj : observation object

Output Messages:

1. 'EXPERIMENT for the given identifier is not found'
2. 'OBSERVATION for the given identifier is not found'
3. Quantity of the particular observation retrieved from the data base will be shown on the screen for each unit in the layout in the order of the layout of the units and blocks with total quantity calculated for each block.
4. Total quantity of the blocks will be displayed through a bar diagram.
5. The quantity of each particular observation for each treatment combination in each block will be displayed in a two dimensional table along with the sum of the quantities for each treatment combination and each block. If there are missing observations, their values are estimated first and then the table is presented.

6. Quantities of the treatment combinations will be expressed through a bar diagram.
7. Analysis of Variance Table will be displayed.

Operation: From the first two input parameters, the function finds the experiment and through the experiment it looks for the design name and the *a priori* layout information from the data base. Once the function receives from the user the object of the observation (o.obj), it goes and searches for all the particular observation quantities (op.qnty) for all the units of the experiment and the message 3 is displayed. If the function is not able to find the required data, then either the message 1 or 2 will appear on the screen. Then follows the message 4. Afterwards, the function sorts out the data in the order of the treatment combinations and blocks and the message 5 appears. After this, the message 6 is displayed on the screen. Finally, the function calculates the values of the sum of squares, mean sum of squares and the values of the F-distributition and then the message 7 appears. If the function is not able to find the required data, then either the message 1 or the message 2 will appear on the screen. Now we present the model and the formulæ to compute the values of the sum of squares and the contents of the analysis of variance table.

The **model** for Randomized Block Design is :

$$y_{ij} = \mu + b_i + t_j + e_{ij}$$

where y_{ij} : yield of the i^{th} block and j^{th} treatment;

e_{ij} : normally and independently distributed around a mean of zero with variance σ^2;

b : number of blocks;

t : number of treatments;

i = 1, 2, ..., r;

j = 1, 2, ..., t.

The yields of the experiment may be arranged in an $r * t$ table.

Treatment

Block	1	2	3	...	t	
1	y_{11}	y_{12}	etc.			$Y_{1.}$
2						$Y_{2.}$
3						$Y_{3.}$
.						.
.						.
.						.
r						$Y_{r.}$
	$Y_{.1}$	$Y_{.2}$	$Y_{.3}$		$Y_{.t}$	$Y_{..}$

ANALYSIS OF VARIANCE TABLE FOR R.B.D.

SOURCE	D.F.	SUM OF SQUARES
Blocks	r - 1	$\sum_i \frac{Y^2_{i.}}{t} - \frac{Y^2_{..}}{rt}$
Treatments	t - 1	$\sum_j \frac{Y^2_{.j}}{r} - \frac{Y^2_{..}}{rt}$
Error	(r - 1) (t - 1)	By subtraction
Total	rt - 1	$\sum y^2_{ij} - \frac{Y^2_{..}}{rt}$

The mean squares are obtained by dividing the sum of squares by the corresponding degrees of freedom. The calculated values of the F-distribution is obtained by dividing the block mean sum of squares and the treatment mean sum of squares by the error mean sum of squares.

8.2. FUNCTION: **MIXED FACTORIAL DESIGN ANALYSIS**

Aim and input parameters are the same as for the first function for Randomized Complete Block Design Analysis (cf 8.1.). In addition to the **output messages** mentioned in the first function, the condensed tables are formed and displayed on the screen. Now we present the model and the formulæ to compute the values of the sum of squares and the contents of the analysis of variance table.

The **model** for Mixed Factoriel Design for three factors is :

$$y_{ijkl} = \mu + b_i + c_j + d_k + (cd)_{jk} + f_l + (df)_{kl} + (cf)_{jl} + (cdf)_{jkl} + e_{ijkl}$$

where y_{ijkl} : yield of the i^{th} block, belonging to the j^{th} level of the first factor, k^{th} level of second factor and l^{th} level of the third factor;

μ : overall mean effect;

b_i : effect due to the i^{th} block;

c_j: main effect due to the j^{th} level of the first factor;

d_k : main effect due to the k^{th} level of the second factor;

f_l : main effect due to the l^{th} level of the third factor;

$(cd)_{jk}$: interaction effect due to the j^{th} level of the first factor and k^{th} level of the second factor;

$(df)_{kl}$: interaction effect due to the k^{th} level of the second factor and l^{th} level of the third factor;

$(cf)_{jl}$: interaction effect due to the j^{th} level of the first factor and l^{th} level of the third factor;

$(cdf)_{jkl}$: interaction effect due to the j^{th} level of the first factor, k^{th} level of the second factor and l^{th} level of the third factor;

e_{ijkl} : random error, normally and independently distributed with mean zero and variance σ^2.

The **formulæ** for sum of squares are as follows :

Let $t = cdf$;

C.F : correction factor $= \frac{Y^2_{...}}{bt}$;

BSS : Block Sum of Squares $= \sum_i \frac{Y^2_{i...}}{t} - C.F$;

M1SS : Main effect of the first factor = $\sum_{j} \frac{Y^2_{.j..}}{dfb}$ - C.F ;

M2SS : Main effect of the 2nd factor = $\sum_{k} \frac{Y^2_{..k.}}{cfb}$ - C.F ;

M3SS : Main effect of the 3rd factor = $\sum_{l} \frac{Y^2_{...l}}{cdb}$ - C.F ;

RINT12 : Raw interaction effect between first factor and second factor

$$= \sum_{jk} \frac{Y^2_{.jk.}}{fb} - C.f ;$$

RINT23 : Raw interaction effect between second factor and third factor

$$= \sum_{kl} \frac{Y^2_{..kl}}{cb} - C.F ;$$

RINT13 : Raw interaction between first factor and second factor

$$= \sum_{jl} \frac{Y^2_{.j.l}}{db} - C.F ;$$

INT12 : interaction effect between first factor and second factor

$$= RINT12 - M1SS - M2SS ;$$

INT23 : interaction effect between second factor and third factor

$$= RINT23 - M2SS - M3SS ;$$

INT13 : interaction effect between first factor and third factor

$$= RINT13 - M1SS - M3SS ;$$

RINT123 : Raw interaction effect between first factor, second factor and third factor = $\sum_{jkl} \frac{Y^2_{.jkl}}{b}$ - C.F ;

INT123 : interaction effect between first factor, second factor and third factor

$$= RINT123 - (M1SS + M2SS + M3SS) ;$$

TSS : Total sum of squares = $\sum_{ijkl} Y^2_{ijkl}$ - C.F .

ANALYSIS OF VARIANCE TABLE FOR MIXED FACTORIAL DESIGN WITH 3 FACTORS

SOURCE	D.F.	SUM OF SQUARES
Block	(b - 1)	BSS
Main effect C	(c - 1)	M1SS
Main effect D	(d - 1)	M2SS
Interaction effect CD	(c - 1) (d - 1)	INT12
Main effect F	(f - 1)	M3SS
Interaction effect DF	(d - 1) (f - 1)	INT23
Interaction effect CF	(c - 1) (f - 1)	INT13
Interaction CDF	(c - 1) (d - 1) (f - 1)	INT123
Error	(b - 1) (t - 1)	by subtraction
Total	bt - 1	TSS

The mean sum of squares are obtained by dividing the sum of squares by the corresponding degrees of freedom. The calculated value of the F-distribution is obtained by dividing each of the first eight mean sum of squares by the error mean sum of squares.

8.3. FUNCTION: **SPLIT-PLOT DESIGN ANALYSIS**

Aim and input parameters are the same as for the first function for Randomized Complete Block Design Analysis (cf 8.1.). In addition to the **output messages** mentioned in the first function, the condensed tables are formed and displayed on the screen. Now we present the model and the formulæ to compute the values of the sum of squares and the contents of the analysis of variance table.

The **model** for a Split-Plot design is :

$$y_{ijk} = \mu + r_i + t_j + (rt)_{ij} + s_k + (ts)_{jk} + e_{ijk}$$

where y_{ijk} : Yield of i^{th} replicate for j^{th} whole-plot treatment and for k^{th} split-plot treatment;

$(rt)_{ij}$: normally and independently distributed with mean zero and variance σ^2;

e_{ijk} : normally and independently distributed with mean zero and variance σ^2;

$i = 1, 2, \ldots r$;

$j = 1, 2, \ldots t$;

$k = 1, 2, \ldots s$;

r = number of replicates (blocks);

t = number of whole-plot treatments;

s = number of split-plot treatments.

ANALYSIS OF VARIANCE TABLE FOR SPLIT-PLOT DESIGN

SOURCE	D.F.	SUM OF SQUARES
Replicates R	r - 1	$\sum_{i} \frac{Y^2_{i..}}{ts} - \frac{Y^2_{...}}{rts}$
Whole-plot treatments T	t - 1	$\sum_{j} \frac{Y^2_{.j.}}{rs} - \frac{Y^2_{...}}{rts}$
R x T	(r - 1) (t - 1)	$\sum_{ij} \frac{Y^2_{ij.}}{s} - \sum_{i} \frac{Y^2_{i..}}{ts} - \sum_{j} \frac{Y^2_{.j.}}{rs} + \frac{Y^2_{...}}{rts}$
Split-plot treatments S	s - 1	$\sum_{k} \frac{Y^2_{..k}}{rt} - \frac{Y^2_{...}}{rts}$
S x T	(t - 1) (s - 1)	$\sum_{jk} \frac{Y^2_{.jk}}{r} - \sum_{j} \frac{Y^2_{.j.}}{rs} - \sum_{k} \frac{Y^2_{..k}}{rt} + \frac{Y^2_{...}}{rts}$
Error	(r - 1) t (s - 1)	By subtraction
Total	rts - 1	$\sum_{ijk} y^2_{ijk} - \frac{Y^2_{...}}{rts}$

Error sum of squares is obtained by subtracting all other sum of squares from the total sum of squares. The mean sum of squares are obtained by dividing each sum of squares by corresponding degrees of freedom D.F. The calculated values of the F-distribution is obtained by dividing each of the first five mean sum of squares by the error mean sum of squares. If we take $(rt)_{ij}$ as error term, then the calculated value of F distribution can be obtained by dividing the whole plot treatment mean sum of squares by the R x T mean sum of squares.

CHAPTER 9

DATA BASE MANAGEMENT MODULE

In this chapter each procedure of the data base primitives of the *data base management module* is presented. First the update procedures, namely, creation, modification and deletion of entities are explained and then the necessary procedures for the access to the data from the data base.

9.1. CREATION

9.1.1. EXPERIMENT CREATION

This procedure creates an EXPERIMENT entity in the data base. It provides a unique experiment number as identifier for the occurrence of this entity.

Input Parameters:

exp.stdate: experiment starting date
exp.endate: experiment ending date
exp.objective: experiment objective

Output Messages:

If the operation succeeds, then an occurrence of the entity type EXPERIMENT, which has just been created, and the message code denoting the success appear. Otherwise, only the message code of the failure does.

9.1.2. DESIGN CREATION

This procedure creates a DESIGN entity. Before creation, it finds out if an entity DESIGN already exists with the same name. If it exists, then the procedure simply inserts an access path ED between DESIGN and the proper EXPERIMENT identified. Otherwise, it creates a new entity DESIGN and inserts an access path ED.

Input Parameters:

d.name : design name
d.desc : design description
exp.num : experiment number

Output Messages:

If the operation succeeds, then an entity variable referring to the DESIGN which has just been created and the message code denoting the success appear. Otherwise, only the message code of failure does.

9.1.3. STATION CREATION

This procedure creates a STATION entity together with its associated SITE entities. Before creation, the procedure makes sure that another station with the same name does not exist. A unique site number is provided and an access path SS is inserted for each site by the procedure.

Input Parameters:

st.name: station name
st.adr: station address
total number of sites
for each site
    s.type: experimental site type
    s.size: experimental site size

Output Messages:

If the operation succeeds, then an entity variable referring to the STATION, which has just been created, and the message code denoting the success appear. Otherwise, only the message code of failure does.

9.1.4. SITE CREATION

This procedure creates a SITE entity in the data base and connects it with the already existing station. It provides a site number as identifier.

Input Parameters:

st.name : station name
s.type : experimental site type
s.size : experimental site size

Output Messages:

If the operation succeeds, then an entity variable referring to the SITE which has just been created and the message code denoting the success appear. Otherwise, only the message code of failure does.

9.1.5. CLIMATE CREATION

This procedure creates a CLIMATE entity and inserts an access path SC.

Input Parameters:

st.name : station name
cl.date : climate observation date
rain : quantity of rain in mm
t.max : maximum temperature in °C
t.min : minimum temperature in °C
rad : radiation in joules/square cm

Output Messages:

Either it returns an entity variable representing the newly created entity CLIMATE with the message code indicating success or it returns a message code of failure.

9.1.6. BLOCK CREATION

This procedure creates a BLOCK entity together with its associated UNIT entities and connects each BLOCK entity with the corresponding EXPERIMENT entity and SITE entity by creating the OCCUPY entity and access paths BY, SY and EY. Unique identifier numbers for block and units are provided by the procedure.

Input Parameters:

exp.num : experiment number
s.num : site number
b.rnum : block row number
b.cnum : block column number
b.type : block type
total number of units in this block
for each unit
    u.rnum: unit row number
    u.cnum: unit column number

Output Messages:

If an occurrence of the entity BLOCK has been created in the data base, then the procedure returns an entity variable representing the newly created BLOCK entity along with a message code of success. Otherwise, it returns a message code to indicate the failure.

9.1.7. OCCUPY CREATION

This procedure is useful when a particular block belonging to a certain site is used by more than one experiment. It creates an OCCUPY entity and connects it with the BLOCK, SITE and EXPERIMENT concerned with their respective access paths BY, SY and EY.

Input Parameters:

exp.num : experiment number
s.num : site number
b.num : block number

Output Messages:

If an occurrence of the entity OCCUPY has been created in the data base, then the procedure returns an entity variable representing the newly created OCCUPY entity along with a message code of success. Otherwise, it returns a message code to indicate the failure.

9.1.8. TREATMENT CREATION

This procedure creates a TREATMENT entity. Before creation, it makes sure that another entity of the same type does not exist with the same treatment name in the data base.

Input Parameters:

trt.name : treatment name

Output Messages:

If the operation succeeds, then an entity variable referring to the TREATMENT, which has just been created, and the message code denoting success appear. Otherwise, only the message code of failure does.

9.1.9. OBSERVATION CREATION

This procedure creates an OBSERVATION entity. Before creation, it makes sure that another entity of the same type does not exist with the same observation object in the data base.

Input Parameters:

o.obj : object of observation

Output Messages:

If the operation succeeds, then an entity variable referring to the OBSERVATION, which has just been created, and the message code denoting success appear. Otherwise, only the message code of failure does.

9.1.10. PROBLEM CREATION

This procedure creates a PROBLEM entity. Before creation, it makes sure that another entity of this type does not exist with the same problem name in the data base.

Input Parameters:

p.name : problem name

Output Messages:

If the operation succeeds, then an entity variable referring to the PROBLEM, which has just been created, and the message code denoting success appear. Otherwise, only the message code of failure does.

9.1.11. PARTICULAR TREATMENT CREATION

This procedure creates a TRT-PARTIC entity. Before creation, it checks that there does not exist another entity of the same type with the same identifier values. The first level of identification is done through the entities EXPERIMENT, TREATMENT and UNIT, and the second level by the attribute values of date and time of the entity TRT-PARTIC. This procedure also creates the access paths ET, UT and TP.

Input Parameters:

exp.num : experiment number
trt.name : treatment name
u.num : unit number
tp.date : particular treatment date
tp.time : particular treatment time
tp.qnty : particular treatment quantity
tp.meas : measure used for the quantity
tp.level : particular treatment level
tp.type : particular treatment type
tp.opertype : particular treatment operation type

Output Messages:

If an occurrence of the entity TRT-PARTIC has been created in the data base, then the procedure returns an entity variable representing the newly created TRT-PARTIC entity along with a message code of success. Otherwise, it returns a message code to indicate the failure.

9.1.12. GLOBAL TREATMENT CREATION

This procedure creates a TRT-GLOBAL entity . Before creation, it checks that there does not exist another entity of the same type with the same identifier values. The first level of identification is done through the entities EXPERIMENT and TREATMENT and the second level by the attribute values of date and time of the entity TRT-GLOBAL. This procedure also creates the access paths EG and TG.

Input Parameters:

exp.num : experiment number
trt.name : treatment name
tg.date : global treatment date
tg.time : global treatment time
tg.qnty : global treatment quantity
tg.meas : measure used for the quantity
tg.opertype : global treatment operation type

Output Messages:

If an occurrence of the entity TRT-GLOBAL has been created in the data base, then the procedure returns an entity variable representing the newly created

TRT-GLOBAL entity along with a message code of success. Otherwise, it returns a message code to indicate the failure.

9.1.13. PARTICULAR OBSERVATION CREATION

This procedure creates an OBS-PARTIC entity. Before creation, it checks that there does not exist another entity of the same type with the same identifier values. The first level of identification is done through the entities EXPERIMENT, OBSERVATION and UNIT, and the second level by the attribute values of date and time of the entity OBS-PARTIC. This procedure also creates the access paths EP, UO and OP.

Input Parameters:

exp.num : experiment number
o.obj : object of observation
u.num : unit number
op.date : particular observation date
op.time : particular observation time
op.qnty : particular observation quantity
op.meas : measure used for the quantity
op.level : particular observation level
op.method : particular observation method

Output Messages:

If an occurrence of the entity OBS-PARTIC has been created in the data base, then the procedure returns an entity variable representing the newly created OBS-PARTIC entity along with a message code of success. Otherwise, it returns a message code to indicate the failure.

9.1.14. GLOBAL OBSERVATION CREATION

This procedure creates a OBS-GLOBAL entity. Before creation, it checks that there does not exist another entity of the same type with the same identifier values. The first level of identification is done through the entities EXPERIMENT and OBSERVATION, and the second level by attribute values of date and time of entity OBS-GLOBAL. This procedure also creates the access paths EO and OG.

Input Parameters:

exp.num : experiment number
o.obj : object of observation
og.date : global observation date
og.time : global observation time
og.qnty : global observation quantity
og.meas : measure used for the quantity
og.method : global observation method

Output Messages:

If an occurrence of the entity OBS-GLOBAL has been created in the data base, then the procedure returns an entity variable representing the newly created OBS-GLOBAL entity along with a message code of success. Otherwise, it returns a message code to indicate the failure.

9.1.15. PARTICULAR PROBLEM CREATION

This procedure creates a PR-PARTIC entity. Before creation, it checks that there does not exist another entity of the same type with the same identifier values. The first level of identification is done through the entities EXPERIMENT, PROBLEM and UNIT, and the second level by the attribute values of date and time of the entity PR-PARTIC. This procedure also creates the access paths ER, UR and PR.

Input Parameters:

exp.num : experiment number
u.num : unit number
p.name : problem name
pp.date : particular problem date
pp.time : particular problem time

Output Messages:

If an occurrence of the entity PR-PARTIC has been created in the data base, then the procedure returns an entity variable representing the newly created PR-PARTIC entity along with a message code of success. Otherwise, it returns a message code to indicate the failure.

9.1.16. GLOBAL PROBLEM CREATION

This procedure creates a PR-GLOBAL entity. Before creation, it checks that there does not exist another entity of the same type with the same identifier values. The first level of identification is done through the entities EXPERIMENT and PROBLEM, and the second level by attribute values of date and time of the entity PR-GLOBAL. This procedure also creates the access paths EL and PL.

Input Parameters:

exp.num : experiment number
p.name : problem name
pg.date : global problem date
pg.time : global problem time

Output Messages:

If an occurrence of the entity PR-GLOBALhas been created in the data base, then the procedure returns an entity variable representing the newly created PR-GLOBAL entity along with a message code of success. Otherwise, it returns a message code to indicate the failure.

9.2. MODIFICATION

For each entity to be modified, *all* its attribute values are entered, namely, the attributes values that are to be modified as well as those that do not require any modification.

9.2.1. EXPERIMENT MODIFICATION

This procedure modifies the values of the attributes of the entity type EXPERIMENT. Since the identifier value of the site is provided by the procedure of creation of experiment respecting its uniqueness, only other attribute values may need modification.

Input Parameters:

exp.num : experiment number
exp.stdate : experiment starting date
exp.endate : experiment ending date
exp.objective : experiment objective

Output Messages:

If the entity is modified in the data base, then the modified entity type EXPERIMENT along with a message code to indicate success appears. Otherwise, a code number indicating the cause of non modification appears.

9.2.2. DESIGN MODIFICATION

This procedure modifies the desired values of the attributes of the entity type DESIGN. Before proceeding to the modification, it verifies that there does not exist another design with the new identifier value of d.name in the data base. This verification is done only if the original identifier value of the design is to be changed.

Input Parameters:

od.name : old design name
nd.name : new design name
d.desc : design description

Output Messages:

If the entity is modified in the data base, then the modified entity type DESIGN along with a message code to indicate the success appears. Otherwise, a code indicating the cause of non modification appears.

9.2.3. STATION MODIFICATION

This procedure modifies the desired values of the attributes of the entity type STATION. Before proceeding to the modification, it verifies that there does not exist another station with the new identifier value of st.name in the data base. This verification is done only if the original identifier value of the station is to be changed.

Input Parameters:

ost.name : original station name
nst.name : new station name
st.adr : station address

Output Messages:

If the entity is modified, then the modified entity type STATION along with a message code to indicate the success appears. Otherwise, a code does, indicating the cause of non modification.

9.2.4. SITE MODIFICATION

This procedure modifies the values of the attributes of the entity type SITE. Since the identifier value of the site is provided by the procedure for creation of site respecting its uniqueness, only other attribute values may need modification.

Input Parameters:

s.num : site number
s.type : site type
s.size : site size

Output Messages:

If the entity is modified, then the modified entity type SITE along with a message code to indicate the success appears. Otherwise, a code does, indicating the cause for non modification.

9.2.5. CLIMATE MODIFICATION

This procedure modifies the entity type CLIMATE. Since the cl.date is an identifier within a station, the procedure first checks if there exists any other entity type CLIMATE with the new value of the date. This check is done only if the new date is different from the original date.

Input Parameters:

st.name : station name
ocl.date : original climate date (this date is for identifying the entity CLIMATE)
ncl.date : new climate date
rain : quantity of rain in mm
t.max : maximum temperature °C
t.min : minimum temperature °C
rad : radiation in joules/square cm

Output Messages:

If the entity is modified, then the modified entity type CLIMATE along with a message code to indicate the success appears. Otherwise a code indicating the cause for non modification appears.

9.2.6. BLOCK MODIFICATION

This procedure modifies the values of the attributes of the entity type BLOCK. Since the identifier value of the site is provided by the procedure for creation of BLOCKs respecting its uniqueness, only other attribute values may need modification.

Input Parameters:

b.num : block number
b.rnum : block row number
b.cnum : block column number
b.type : block type

Output Messages:

If the entity is modified, then the modified entity type BLOCK along with a message code to indicate the success appears. Otherwise, a code indicating the cause for non modification appears.

9.2.7. UNIT MODIFICATION

This procedure modifies the values of the attributes of the entity type UNIT. Since the identifier value of the site is provided by the procedure of creation of UNIT respecting its uniqueness, only other attribute values may need modification.

Input Parameters:

u.num : block number
u.rnum : block row number
u.cnum : block column number

Output Messages:

If the entity is modified, then the modified entity type UNIT along with a message code to indicate the success appear. Otherwise, a code indicating the non modification.

9.2.8. TREATMENT MODIFICATION

This procedure modifies the entity type TREATMENT identified by the old name otrt.name. Before modifying the otrt.name, it checks whether there exists another entity of the same type with the new name ntrt.name. In case of nonexistence of such an entity type TREATMENT in the data base, the procedure modifies the original identifier.

Input Parameters:

otrt.name : old treatment name

ntrt.name : new treatment name

Output Messages:

If the entity is modified in the data base, then the modified entity type TREATMENT along with a message code to indicate the success appears. Otherwise, a code number does, indicating the cause for non modification.

9.2.9. OBSERVATION MODIFICATION

This procedure modifies the entity type OBSERVATION identified by the old name of the object of observation oo.obj. Before modifying the oo.obj, it checks whether there exists another entity of the same type with the new object no.obj. In case of nonexistence of such an entity type OBSERVATION in the data base, the procedure modifies the original identifier.

Input Parameters:

oo.obj : old observation object

no.obj : new observation object

Output Messages:

If the entity is modified in the data base, then the modified entity type OBSERVATION along with a message code to indicate the success appears. Otherwise, a code number does, indicating the cause for non modification.

9.2.10. PROBLEM MODIFICATION

This procedure modifies the entity type PROBLEM identified by op.name. Before modifying the op.name, it checks whether there exists another entity of the same type with the new name np.name. In case of nonexistence of such an entity type PROBLEM in the data base, the procedure modifies the original identifier.

Input Parameters:

op.name : old problem name

np.name : new problem name

Output Messages:

If the entity is modified in the data base, then the modified entity type PROBLEM along with a message code to indicate the success appears. Otherwise, a code number does, indicating the cause for non modification.

9.2.11. PARTICULAR TREATMENT MODIFICATION

This procedure modifies the values of the attributes of the entity type TRT-PARTIC. If this entity type exists already in the data base for the given new identifier values exp.num, u.num, trt.name,tp.date and tp.time, then no modification is performed. If it does not exist, then the original entity which needs modification is modified with the new values. If the values of the date and time of the original entity and the new date and time are the same, then the modification intended for other attribute values is done.

Input Parameters:

exp.num : experiment number
u.num : unit number
trt.name : treatment name
tp.odate : particular treatment old date
tp.otime : particular treatment old time
tp.ndate : particular treatment new date
tp.ntime : particular treatment new time
tp.qnty : particular treatment quantity
tp.meas : measure used for the quantity
tp.level : particular treatment level
tp.type : particular treatment type
tp.opertype : particular treatment operation type

Output Messages:

If the entity is modified in the data base, then the modified entity type TRT-PARTIC along with a message code to indicate the success appears. Otherwise, a code number does, indicating the cause for non modification.

9.2.12. GLOBAL TREATMENT MODIFICATION

This procedure modifies the values of the attributes of the entity type TRT-GLOBAL. If this entity type exists already in the data base for the given new identifier values exp.num, trt.name, tg.date and tg.time, then no modification is performed. If it does not

exist, then the original entity which needs modification is modified with the new values. If the values of the date and time of the original entity and the new date and time are the same, then the modification intended for other attribute values is done.

Input Parameters:

exp.num : experiment number
trt.name : treatment name
tg.odate : global treatment old date
tg.otime : global treatment old time
tg.ndate : global treatment new date
tg.ntime : global treatment new time
tg.qnty : global treatment quantity
tg.meas : measure used for the quantity
tg.opertype : global treatment operation type

Output Messages:

If the entity is modified in the data base, then the modified entity type TRT-GLOBAL along with a message code to indicate the success appears. Otherwise, a code number does, indicating the cause for non modification.

9.2.13. PARTICULAR OBSERVATION MODIFICATION

This procedure modifies the values of the attributes of the entity type OBS-PARTIC. If this entity type exists already in the data base for the given new identifier values exp.num, u.num, o.obj, op.date and op.time, then no modification is performed. If it does not exist, then the original entity which needs modification is modified with the new values. If the values of the date and time of the original entity and the new date and time are the same, then the modification intended for other attribute values is done.

Input Parameters:

exp.num : experiment number
u.num : unit number
o.obj : observation object
op.odate : particular observation old date
op.otime : particular observation old time
op.ndate : particular observation new date
op.ntime : particular observation new time
op.qnty : particular observation quantity

op.meas : measure used for the quantity
op.level : particular observation level
op.method : particular observation method

Output Messages:

If the entity is modified in the data base, then the modified entity type OBS-PARTIC along with a message code to indicate the success appears. Otherwise, a code number does, indicating the cause for non modification.

9.2.14. GLOBAL OBSERVATION MODIFICATION

This procedure modifies the values of the attributes of the entity type OBS-GLOBAL. If this entity type exists already in the data base for the given new identifier values exp.num, o.obj, op.date and op.time, then no modification is performed. If it does not exist, then the original entity which needs modification is modified with the new values. If the values of the date and time of the original entity and the new date and time are the same, then the modification intended for other attribute values is done.

Input Parameters:

exp.num : experiment number
o.obj : observation object
og.odate : global observation old date
og.otime : global observation old time
og.ndate : global observation new date
og.ntime : global observation new time
og.qnty : global observation quantity
og.meas : measure used for the quantity
og.method : global observation method

Output Messages:

If the entity is modified in the data base, then the modified entity type OBS-GLOBAL along with a message code to indicate the success appears. Otherwise, a code number does, indicating the cause for non modification.

9.2.15. PARTICULAR PROBLEM MODIFICATION

This procedure modifies the values of the attributes of the entity type PR-PARTIC. If this entity type exists already in the data base for the given new identifier values exp.num, u.num, p.name, pp.date and pp.time, then no modification is performed.

Input Parameters:

exp.num : experiment number
u.num : unit number
p.name : problem name
pp.ndate : particular problem new date
pp.odate : particular problem old date
pp.ntime : particular problem new time
pp.otime : particular problem old time

Output Messages:

If the entity is modified in the data base, then the modified entity type PR-PARTIC along with a message code to indicate the success appears. Otherwise, a code number does, indicating the cause for non modification.

9.2.16. GLOBAL PROBLEM MODIFICATION

This procedure modifies the values of the attributes of the entity type PR-GLOBAL. If this entity type exists already in the data base for the given new identifier values exp.num, p.name, pg.date and pg.time, then no modification is performed.

Input Parameters:

exp.num : experiment number
p.name : problem name
pg.odate : global problem old date
pg.otime : global problem old time
pg.ndate : global problem new date
pg.ntime : global problem new time

Output Messages:

If the entity is modified in the data base, then the modified entity type PR-GLOBAL along with a message code to indicate the success appears. Otherwise, a code number does, indicating the cause for non modification.

9.3. DELETION

9.3.1. STATION DELETION

This procedure deletes from the data base the STATION entity identified by the station name st.name together with the sites it possesses and the climate entity it has created. Before deletion, it makes sure that no experiment occupies any site belonging to this station.

Input Parameters:

st.name : station name

Output Message:

A message code indicating the result of the operation appears.

9.3.2. CLIMATE DELETION

This procedure deletes a CLIMATE entity from the data base. A CLIMATE entity is identified by cl.date within a given station.

Input Parameters:

st.name : station name

cl.date : climate date

Output Message:

A message code indicating the result of the operation appears.

9.3.3. SITE DELETION

This procedure deletes a SITE entity from the data base identified by site number s.num.

Input Parameters:

s.num : site number

Output Message:

A message code indicating the result of the operation appears.

9.3.4. BLOC DELETION

This procedure destroys the BLOCK entity identified by its number b.num with all its subordinate UNITS from the data base. Before destruction, it checks that this block is not used by other experiments at the same time as the one from which it was to be removed and it also checks that no unit of this block is linked to the entity TRT-PARTIC, PR-PARTIC or OBS-PARTIC. When a BLOCK is destroyed or its connection to a particular experiment is removed, the entity OCCUPY is also deleted.

Input Parameters:

b.num : block number

exp.num : experiment number

Output Message:

A message code indicating the result of the operation appears.

9.3.5. PARTICULAR TREATMENT DELETION 1

This procedure deletes *all* the TRT-PARTIC entities belonging to an experiment from the data base.

Input Parameters:

exp.num : experiment number

Output Message:

A message code indicating the result of the operation appears.

9.3.6. GLOBAL TREATMENT DELETION 1

This procedure deletes all the TRT-GLOBAL entities belonging to an experiment from the data base.

Input Parameters:

exp.num : experiment number

Output Message:

A message code indicating the result of the operation appears.

9.3.7. PARTICULAR OBSERVATION DELETION 1

This procedure deletes all the OBS-PARTIC entities belonging to an experiment from the data base.

Input Parameters:

exp.num : experiment number

Output Message:

A message code indicating the result of the operation appears.

9.3.8. GLOBAL OBSERVATION DELETION 1

This procedure deletes all the OBS-GLOBAL entities belonging to an experiment from the data base.

Input Parameters:

exp.num : experiment number

Output Message:

A message code indicating the result of the operation appears.

9.3.9. PARTICULAR PROBLEM DELETION 1

This procedure deletes all the PR-PARTIC entities belonging to an experiment from the data base.

Input Parameters:

exp.num : experiment number

Output Message:

A message code indicating the result of the operation appears.

9.3.10. GLOBAL PROBLEM DELETION 1

This procedure deletes all the PR-GLOBAL entities belonging to an experiment from the data base.

Input Parameters:

exp.num : experiment number

Output Message:

A message code indicating the result of the operation appears.

9.3.11. TREATMENT DELETION

This procedure deletes a TREATMENT entity from the data base. Before deletion, it makes sure that this entity is not linked to some other TRT-PARTIC or TRT-GLOBAL entity. The TREATMENT entity is identified by trt.name.

Input Parameters:

trt.name : treatment name

Output Message:

A message code indicating the result of the operation appears.

9.3.12. OBSERVATION DELETION

This procedure deletes an OBSERVATION entity from the data base. Before deletion, it makes sure that this entity is not linked to some other OBS-PARTIC or OBS-GLOBAL entity. The OBSERVATION entity is identified by o.obj.

Input Parameters:

o.obj : observation object

Output Message:

A message code indicating the result of the operation will be passed on to the calling program.

9.3.13. PROBLEM DELETION

This procedure deletes a PROBLEM entity from the data base. Before deletion, it makes sure that this entity is not linked to some other PR-PARTIC or PR-GLOBAL entity. The PROBLEM entity is identified by p.name.

Input Parameters:

p.name : problem name

Output Message:

A message code indicating the result of the operation will be passed on to the calling program.

9.3.14. DESIGN DELETION

This procedure deletes a DESIGN entity from the data base. Before deletion, it makes sure that this entity is not linked to an EXPERIMENT entity. DESIGN entity is identified by d.name.

Input Parameters:

d.name : design name

Output Message:

A message code indicating the result of the operation appears.

9.3.15. EXPERIMENT DELETION

This procedure deletes an EXPERIMENT entity from the data base. Before deletion, it checks that a given EXPERIMENT entity is not linked by an access path to any one of the following entities: OCCUPY, DESIGN, TRT-PARTIC, TRT-GLOBAL, OBS-PARTIC, OBS-GLOBAL, PR-PARTIC and PR-GLOBAL. An EXPERIMENT entity is identified by its number.

Input Parameters:

exp.num : experiment number

Output Message:

A message code indicating the result of the operation appears.

9.3.16. PARTICULAR TREATMENT DELETION 2

This procedure deletes *one* TRT-PARTIC entity *at a time* from the data base. This entity is identified by its attributes date and time after passing through the identifiers of the entities EXPERIMENT, UNIT and TREATMENT.

Input Parameters:

exp.num : experiment number
u.num : unit number
trt.name : treatment name
tp.date : particular treatment date
tp.time : particular treatment time

Output Message:

A message code indicating the result of the operation will be passed on to the calling program.

9.3.17. GLOBAL TREATMENT DELETION 2

This procedure deletes one TRT-GLOBAL entity at a time from the data base. This entity is identified by its attributes date and time after passing through the identifiers of the entities EXPERIMENT and TREATMENT.

Input Parameters:

exp.num : experiment number
trt.name : treatment name
tg.date : global treatment date
tg.time : global treatment time

Output Message:

A message code indicating the result of the operation appears.

9.3.18. PARTICULAR OBSERVATION DELETION 2

This procedure deletes one OBS-PARTIC entity at a time from the data base. This entity is identified by its attributes date and time after passing through the identifiers of the entities EXPERIMENT, UNIT and OBSERVATION.

Input Parameters:

exp.num : experiment number
u.num : unit number
o.obj : observation object
op.date : particular observation date

op.time : particular observation time

Output Message:

A message code indicating the result of the operation will be passed on to the calling program.

9.3.19. GLOBAL OBSERVATION DELETION 2

This procedure deletes one OBS-GLOBAL entity at a time from the data base. This entity is identified by its attributes date and time through the entities EXPERIMENT and OBSERVATION.

Input Parameters:

exp.num : experiment number
o.obj : observation object
og.date : global observation date
og.time : global observation time

Output Message:

A message code indicating the result of the operation will be passed on to the calling program.

9.3.20. PARTICULAR PROBLEM DELETION 2

This procedure deletes one PR-PARTIC entity at a time from the data base. This entity is identified by its attributes date and time through the entities EXPERIMENT, UNIT and PROBLEM.

Input Parameters:

exp.num : experiment number
u.num : unit number
p.name : problem name
pp.date : particular problem date
pp.time : particular problem time

Output Message:

A message code indicating the result of the operation will be passed on to the calling program.

9.3.21. GLOBAL PROBLEM DELETION 2

This procedure deletes one PR-GLOBAL entity at a time from the data base. This entity is identified by its attributes date and time through the entities EXPERIMENT and PROBLEM.

Input Parameters:

exp.num : experiment number
p.name : problem name
pg.date : global problem date
pg.time : global problem time

Output Message:

A message code indicating the result of the operation appears.

9.4. DATA RETRIEVAL

9.4.1. SITE OCCUPATION RETRIEVAL

This procedure identifies a station by its name st.name and finds out all the sites occupied by experiments in that station.

Input Parameter:

st.name : station name

Output Messages:

1. s.num : site number
   s.type : site type
   s.size : site size
   for each site the following details of each experiment will be displayed:
   exp.num : experiment number
   exp.stdate : experiment starting date
   exp.endate : experiment ending date
2. If the station is not found, then a message code will be passed on to the calling program.

9.4.2. EXPERIMENT RETRIEVAL

This procedure finds out the details of the EXPERIMENT entity.

Input Parameter:

exp.num : experiment number

Output Messages:

1. exp.stdate : experiment starting date
   exp.endate : experiment ending date
   exp.obj : experiment objective
2. If the experiment is not found, then a message code will be passed on to the calling program.

9.4.3. PARTICULAR TREATMENT RETRIEVAL

This procedure finds out all the attribute values of the entity TRT-PARTIC.

Input Parameter:

exp.num : experiment number
trt.name : treatment name
u.num : unit number

Output Message:

1. tp.date : particular treatment date
   tp.time : particular treatment time
   tp.qnty : particular treatment quantity
   tp.meas : measure used for the quantity
   tp.level : particular treatment level
   tp.type : particular treatment type
   tp.opertype : particular treatment operation type
2. If either the experiment, the unit or the treatment for the given identifier is not found, then a message code will be passed on to the calling program.

9.4.4. GLOBAL TREATMENT RETRIEVAL

This procedure finds out all the attribute values of the entity TRT-GLOBAL.

Input Parameter:

exp.num : experiment number

Output Message:

1. tg.date : global treatment date
   tg.time : global treatment time
   tg.qnty : global treatment quantity
   tg.meas : measure used for the quantity
   tg.opertype : global treatment operation type
   trt.name : treatment name
2. If either the experiment or the treatment for the given identifier is not found, then a message code will be passed on to the calling program.

9.4.5. PARTICULAR OBSERVATION RETRIEVAL

This procedure finds out all the attribute values of the entity OBS-PARTIC.

Input Parameter:

exp.num : experiment number
o.obj : object of observation
u.num : unit number

Output Message:

1. op.date : particular observation date
   op.time : particular observation time
   op.qnty : particular observation quantity
   op.meas : measure used for the quantity
   op.level : particular observation level
   op.method : particular observation method
2. If either the experiment, the unit or the observation for the given identifier is not found, then a message code will be passed on to the calling program.

9.4.6. GLOBAL OBSERVATION RETRIEVAL

This procedure finds out all the attribute values of the entity OBS-GLOBAL.

Input Parameter:

exp.num : experiment number

Output Message:

1. og.date : global observation date
   og.time : global observation time
   og.qnty : global observation quantity
   og.meas : measure used for the quantity
   og.method : global observation method
   o.obj : object of observation
2. If either the experiment or the observation for the given identifier is not found, then a message code will be passed on to the calling program.

9.4.7. PARTICULAR PROBLEM RETRIEVAL

This procedure finds out all the attribute values of the entity PR-PARTIC.

Input Parameter:

exp.num : experiment number

Output Message:

1. pp.date : particular problem date
   pp.time : particular problem time
   p.name : problem name
   u.num : unit number
2. If either the experiment, the unit or the problem for the given identifier is not found, then a message code will be passed on to the calling program.

9.4.8. GLOBAL PROBLEM RETRIEVAL

This procedure finds out all the attribute values of the entity PR-GLOBAL.

Input Parameter:

exp.num : experiment number

Output Message:

1. pg.date : global problem date
   pg.time : global problem time
   p.name : problem name
2. If either the experiment or the problem for the given identifier is not found, then a message code will be passed on to the calling program.

9.4.9. BLOCKS LAYOUT

This procedure retrieves the data concerning the layout of the blocks in the experimental site for a given experiment.

Input Parameters:

exp.num : experiment number
tp.opertype : particular treatment operation type

Output messages:

1. maximum row number (b.rnum) of the blocks
   maximum column number (b.cnum) of the blocks
   maximum row number (u.rnum) of the units in the first block
   maximum column number (u.cnum) of the units in the first block
   names of the treatments (trt.name) allotted to the last unit of the first block
   maximum level (tp.level) of each particular treatment allotted for the units of the first block
   first unit identifier number (u.num)
   direction in which units are numbered (u.rnum of the second unit of the first block)
   number of factors (number of treatments)
   particular treatment levels (tp.level) allotted for each unit
   treatment names (trt.name) allotted for each unit of the first block
2. if either the experiment or the particular treatment is not found, then a message code will be passed on to the calling program

9.4.10. PARTICULAR OBSERVATION QUANTITIES

This procedure retrieves all the particular observation quantities (op.qnty) belonging to each unit and the measure of the quantities (op.meas) of the first unit for a given o.obj from the data base.

Input Parameters:

first unit identification number (u.num) of the given experiment
maximum number of units for that experiment
o.obj : observation object

Output Messages:

1. particular observation quantities (op.qnty) for each unit
2. particular observation measure for the first unit (we assume that the measure is the same for all the tp.qnty for the given o.obj)
3. if either a unit or an observation is not found, a code number will be passed on to the calling program

9.4.11. DESIGN INFORMATION

This procedure finds out the name of the design for a given experiment.

Input Parameter:

exp.num: experiment number

Output Messages:

1. d.name: name of the design
2. if either the experiment or the design is not found, a message code will be passed to the calling program.

CHAPTER 10

GRAPHIC MANAGEMENT AND INPUT/OUTPUT MODULES

10.1. GRAPHIC MANAGEMENT MODULE

This module is used by Statistical Design Layout Module and Statistical Analysis Module. It uses the graphic tool available in Turbo Pascal.

10.1.1. BLOCK LAYOUT

This procedure displays the Statistical Design Layout on the screen with graphical and coloured image to enable the user to identify easily the different treatment combinations (from *a priori* allotment) appearing in different units in each and every block. If the microcomputer does not have a colour monitor adapter and a colour display terminal, then the treatment combination differences among the units will be identified by various fill patterns on monochrome screen.

Input Parameters:

number of blocks in horizontal direction
number of blocks in vertical direction
number of units in horizontal direction in a block
number of units in vertical direction in a block
direction in which the units are numbered
particular treatment levels for each unit (tp.levels)
treatment combination names (trt.name)
experiment number (exp.num)
design name (d.name)

Output Message:

1. Colour and graphic image of the design layout
2. 'EXP. NUMBER'
3. experiment number (exp.num)
4. design name (d.name)
5. treatment combinations

10.1.2. BAR DIAGRAM

This procedure draws a three dimensional bar diagram on the screen. If the microcomputer does not have a colour monitor adapter and a colour display terminal, then the bars are differenciated by various fill patterns in order to identify the differences on monochrome screen.

Input Parameters:

number of bars needed
number of missing data
quantities for each bar
name of each bar
measure of the quantity for the particular observation (op.meas)
experiment number (exp.num)
observation object (o.obj)

Output Messages:

1. Three dimensional bar diagram appears on the screen (with bars in different colours and fill patterns)
2. 'EXPERIMENT NUMBER'
3. experiment number (exp.num)
4. 'missing observations'
5. number of missing data
6. quantities for each bar
7. name of each bar
8. measure of the quantity for the particular observation (op.meas)
9. observation object (o.obj)

10.2. INPUT/OUTPUT MODULE

This module helps the other modules, namely, Information Update Module, Information Retrieval Module, Statistical Design Layout Module, Statistical Analysis Module, Graphic Management Module and Coordinator Module by supplying the input data necessary for the functioning and by displaying the information (sent by the above mentioned modules) on the screen. It uses the terminal tool for better visual representation of data and for normalization of capital and small letters as desired.

CHAPTER 11

PROSPECTS, INTEGRITY AND SECURITY

Statistical methods play an important role in scientific research in different domains of application. Very often, these methods are used by people who have specialised in a certain field of knowledge. The software system presented here can be used for experimentations applying statistical methods in specialised fields like industry, agriculture, animal husbandry and so on. One can say that the methods and techniques followed for the construction of this software help for automation and pave the way towards an expert system.

The most important characteristics of an expert system is that it relies on a large data base of knowledge. Because a large amount of knowledge is so critical to the success of an expert system, the question of how that knowledge is to be represented is critical to the design of the system (RICH, 83).

In the first part, we describe the prospect of this software by presenting the functions that are possible to implement in this system. Since the quantity and quality of the data stored in the data base is very critical for the success of the software, we discuss in the second part the integrity and security of the data base.

11.1. Functions that are possible to implement in this system

11.1.1. Functions proper to the physical subsystem

1. To find the information about the experiments that have been implemented in certain sites;
2. To find the information about the sites that are reserved for experimentation;
3. To find the information about the sites that are available for experimentation;
4. To find the sites in which experiments are conducted at present and the objectives of those experiments;
5. To caculate the rate of occupation of sites;
6. To draw temperature and rain charts for a given period of time from the meteorological information available in the station;
7. To select the days in which the centre received the highest and lowest rainfall, temperature or radiation in a given year.
8. To calculate the mean and standard deviation from the meteorological information;

9. In agricultural and animal husbandry experiments, to find out the problems encountered globally during experimentation or particularly in certain experimental units in adverse weather conditions.

11.1.2. Functions proper to the experiments

1. To find the time and date of a problem that affected any experimental unit or the whole experiment and to detect the possible causes and choose the more probable ones (a problem may occur because of weather conditions, because of the application of certain treatments or it can be inherent in the experimental material and so on);
2. To find the treatments and observations that followed certain problems;
3. To find whether there had been any difference in the yield or efficiency of the experimental units because of the problem that affected them;
4. To find the experiments or units that had been affected by the same sort of problems, to arrive at a valid conclusion for better future planning and to present the necessary steps to prevent such problems;
5. To choose the layout (local control) that has minimised the variation in the experimental material or within blocks by comparing the variation in the yields of different experiments in agriculture and animal husbandry;
6. To draw the fertility gradiant map and to give suggestions for a better layout, size and shape of the units and blocks;
7. To choose the best season for certain types of crops in agriculture by comparing the yields;
8. To find the experiments under the same objective;
9. From the objective, to choose the observation object for retrieval of data for analysis;
10. To predict a possible problem and to propose preventive actions;
11. To predict the prospect of an experiment at different periods;
12. To choose the best treatments that help producing a better quality and quantity;
13. To choose the best products in industries or the best variety of crops in agriculture;
14. To choose the treatments having certain significant interactions among themselves;
15. To test a hypothesis and arrive at valid inferences using statistical tests of significance (like F-test, χ-square test, t-test);
16. To analyse the data through various statistical methods;

17. To add functions for the layout and analysis of many statistical designs like Completely Randomized Design, Latin Square Design, factorial Design and so on.

11.1.3. Functions for standard entities

1. To find a treatment, a design that has been used frequently;
2. To sort out a list of designs in a certain order and display them on the screen;
3. To sort out in a desired order a list of problems that have occurred in the given research centre;
4. To sort out a list of the object of observation in a desired order.

11.1.4. Other general functions

1. To archieve the data;
2. To normalize date and time.

As we see from some of the functions listed above, the data base we have constructed is a treasure of valuable information for decision making process and automation of every activity in the domain of statistical designs of experiments. A program can be constructed to retreive the data from the data base and make valid inferences intelligently like an expert in that field and to proceed from one action to another interactively with the researcher.

11.2. Integrity and security of the data base

The information that can be extracted from the data base is one of the most valuable resources of the enterprise. The decision making process depends on the quality and quantity of information (ATRE, 80) and hence the need for data integrity and security. In the data base management module, we have a certain number of update primitives necessary for each entity in the data base. In addition, we have more primitives in entity deletion sections for the deletion of many entities at a time. This will be by and large sufficient for the time being for efficient updating and for maintaining the integrity of the data base. Now, we shall discuss about the persons who are responsible for the security and the priorities to be followed in the update of data in this data base.

11.2.1. The persons and their roles

For the security and integrity of the data in this system, we foresee the following three categories of persons responsible for the good functioning of the data base and the application program :

1. The user (researcher or research station manager)
2. The system administrator
3. The application programmer

The research station manager should be responsible for the creation and modification of the entities STATION, SITE and CLIMATE. Day to day activities in updating and retreival of data proper to the experiments can be performed by researchers. It is better for the integrity and security of the system that a user adds, modify or deletes entities in consultation with the system administrator. The system administrator is like a coordinator between the application programmer and the user. He should give a proper training to the user for using the system in the right way. The programmer should also, like the user, take the responsibility to safeguard the integrity of the data in the data base.

11.2.2. Update Primitives Priorities

11.2.2.1. Creation

In the creation, we have the following priorities :

Firstly, the creation of the entities in physical subsystem, namely STATION and SITES;

Secondly, the creation of standard entities DESIGN, TREATMENT, PROBLEMS and OBSERVATION;

Thirdly, the creation of EXPERIMENT dependent entities BLOCK, UNIT, TRT-PARTIC, TRT-GLOBAL, PR-PARTIC, PR-GLOBAL, OBS-PARTIC, OBS-GLOBAL and EXPERIMENT.

11.2.2.2. Deletion

For the deletion of data, the following priorities are preferred :

First, the experiment dependent entities are deleted in the following order :

a) TRT-PARTIC, TRT-GLOBAL, PR-PARTIC, PR-GLOBAL, OBS-PARTIC and OBS-GLOBAL;

b) BLOCK, UNIT and EXPERIMENT;

Secondly, the entities of the physical subsystem are deleted in the following order: SITE, CLIMATE and then STATION;

Thirdly, the standard entities are deleted if they are no more needed in the data base.

The primitives designed to delete one entity at a time from among the experiment depending entities can be used by the researchers. The primitives destined to delete many entities proper to an experiment can be performed by the system administrator. They are designed in such a way that a block can be deleted only if it has no connection with any one of the following relations, namely, TRT-PARTIC, PR-PARTIC and OBS-PARTIC. Standard entities in the same way cannot be deleted unless they are free from any connection with other entities. An EXPERIMENT can be deleted only if it has no connection with any other entities in the data base. A SITE can be deleted only if it is not connected through the OCCUPY relation to EXPERIMENT or BLOCK. A STATION can be deleted only if it has no link with any SITE or CLIMATE entities. In this way, the security and integrity of the data base is better maintained.

11.2.3. Access and insertion control

In the application program that has been implemented so far with this data base, we have several measures of control to access and to insert data. Firstly, one has to identify the experiment by an identification number. Secondly, to have the layout displayed, the user needs an experiment identification number and a particular treatment operation type (*a priori* label allotment type). Thirdly, he has to enter the right object of observation to retreive data for analysis. Finally, to create the data of observation, he needs an experiment identification number and the first unit number of his experiment. These numbers had been given to him during the time of designing layout. In fact, he is reminded then to note the information in his diary.

11.2.4. External accidents

Certain unexpected or expected accidents like power cut, run-time errors and program errors may cause a destruction of data. Therefore, it is advisable to use the programs that are tested for their validity in logics, specifications and codes. The programs should be able to detect the errors that cause such accidents. It is better to back up all the data from the data base from time to time.

A better security measure can be planned depending on the future growth of the system and the needs of the application.

CHAPTER 12

CONCLUSION

The software program proposed in this dissertation has integrated the essential qualities of a software, namely, generality, flexibility and extensibility. Though the objective of this dissertation is limited to the implementation of Randomized Block Design, 3x2x2 Mixed Factorial Design and Split-Plot Design for agricultural experiments, the program already implemented can be used for factorial experiments from one to three factors. The conception of the software is such that it can be extended to other similar experiments of many statistical designs and in many other applications like industry, medecine, animal husbandry and so on with minimum modification and low cost. As we have noted in the previous chapter, there are many more useful functions or subsystems that can be added to the existing ones without major changes.

The modularity of the functions facilitates easy modifications, e.g. if we have to modify some functions in the Information Update module, we need not disturb the module for Statistical Analysis. Thus, error detection and correction become easier and less costly. The hierarchical structure of the software system with the relation "use" has also helped to develop the program in separate modules independently. The import/export relation used has improved the interface between the user and the system and has enabled the user to follow the sequence of operations.

As the data base design represents the real environment of the statistical experiments, it satisfies the following objectives:

1. Different functions of the domain of application can be served effectively by the same data base;
2. Consistent information can be supplied for the decision making process;
3. Application programs can be developed, maintained and enhanced faster and more economically;
4. Easier procedure for computer operations can be established.

The conception of the data base is so generalized that it can serve many applications for statistical designs of experiments in agriculture, industry, animal husbandry and so on. It can be used for both complete and incomplete Experimental Designs.

NDBS has helped for the development of this complex data base and it can become more powerful if it is fully developed as expected.

The result of the operation of this software program for the given three designs of experiments in the appendix is an encouraging sign and it proves that it is possible to build this system into a powerful one for automation of this application. The design layout

that is shown in the visual and graphic image will make the work of the researcher easy and pleasant.

I feel that I have made a healthy jump from the objective of this dissertation that has been proposed in the introduciton to the new vision that can bloom into a reality given the present advancement in science and computer technology. It will be my endeavour to continue the work started and develop this software into an "intelligent" and powerful system to meet the requirements of the application of statistical experiments.

REFERENCES

ATRE, 80 — Atre S., *Data Base : Structured Techniques for Design, Performance and Management,* John Wiley & Sons, New York and Singapore, 1980.

DAGNELIE, 81 — Dagnelie P., *Principes d'expérimentation,* Presses Agronomiques de Gembloux, 1981.

HAINAUT, 86 — Hainaut J.L., *Conception assistée des applications informatiques. 2. Conception de la base de données,* Masson, Presses Universitaires de Namur, 1986.

HAINAUT, 87 — Hainaut J.L., *NDBS. A simple data base system for small computers,* Institut d'Informatique des Facultés Universitaires de Namur, 1987.

JACKSON, 75 — Jackson M.A., *Principles of Program Design,* Academic Press, New York, 1975.

KEMP, 73 — Kempthorne O., *The Design and Analysis of Experiments,* Robert E.Krieger Publishing Company, Huntington, New york, 1973.

K.MOORTY, 77 — Krishnamoorty A. and Thainese J., *Effect of fertilizer and spacing on rice yield,* M.Sc. Project Report, Madras Christian College, Madras, 1977.

PANSE, 67 — Panse U.G. and Sukhatme P.U., *Statistical Methods for Agricultural Workers,* Icar, 2nd edition, New Dehli, 1967.

RICH, 83 — Rich E., *Artificial Intelligence* , Mc Graw-Hill, USA, 1983.

APPENDIX

PROGRAM RESULTS

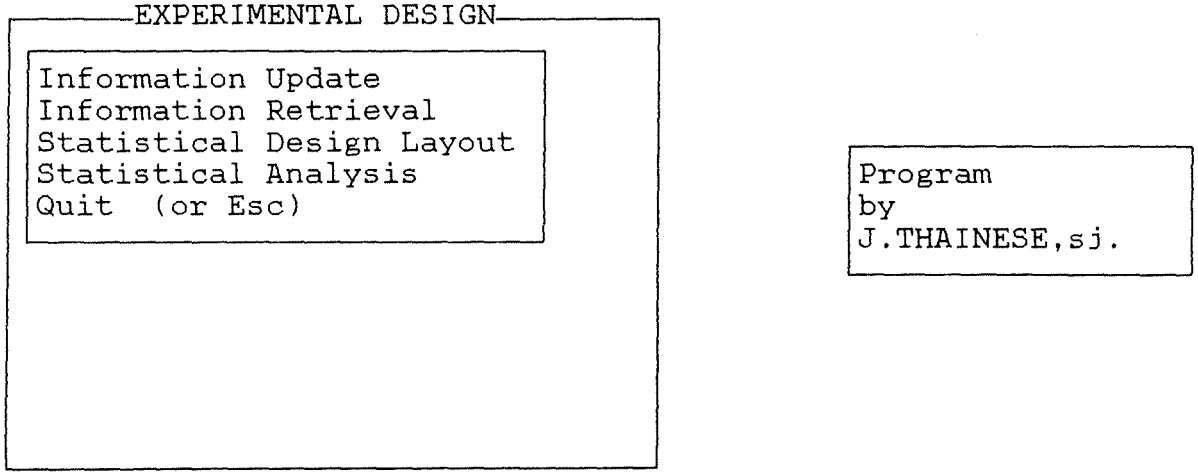
The software program has the following Turbo Pascal Units: utils, windows, gadgets, io_data, creation, mod_data, del_data, graphmod and db. The first three units together form an interface toolkit used for better presentation of the coordinator module. These units had been programmed by D. Pountain (cf. *Personal Computer World* , February 1990) and composed together with certain additions of functions by O. Marchand in the Faculty of Computer Science, Namur. These units are adapted to our program needs. The db unit is the DBMS (NDBS).

To run this program, we need the compiled version of the application program COORDINA. EXE, the data base EXPERIME. DTB for designs of experiments and the Turbo Pascal graphic drivers and fonts for graphic and colour displays.

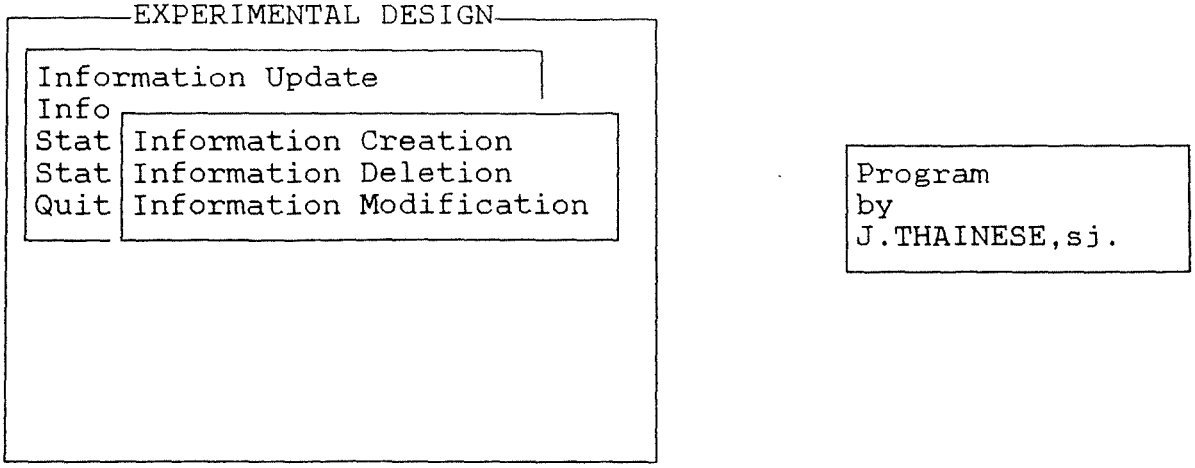
The user is guided during the operation to enter the proper data and work interactively. In the first part, we present the different choices of the functions available to the user (cf. chapter 4), then some of the data retreived from the data base, their statistical layout and analysis for Randomized Complete Block Design, for 3x2x2 Mixed Factorial Design and for Split-Plot Design.

1. Choices of the functions from the different modules

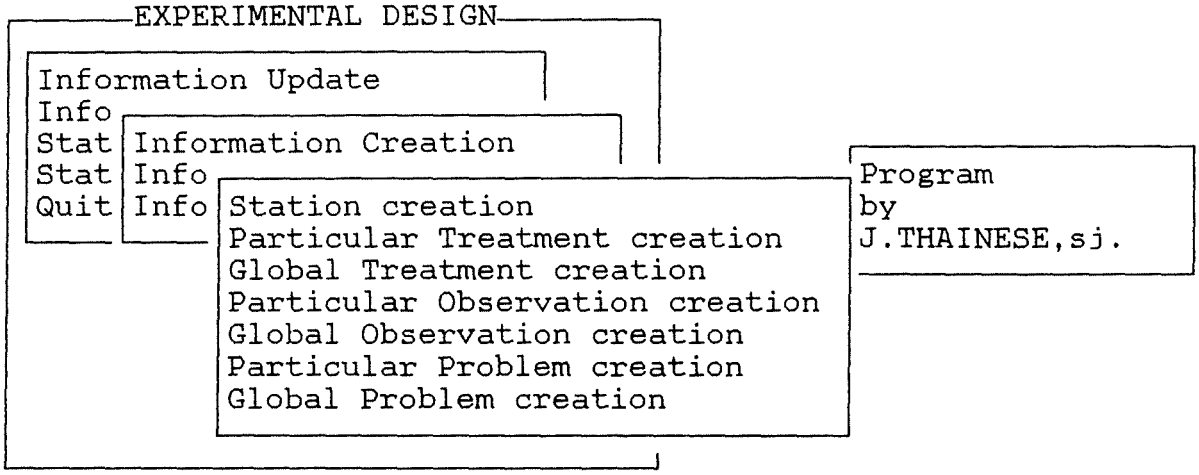
These choices are displayed in colours with background shadows on the screen. If the colour printer is available for text colour, they can be printed in colour.



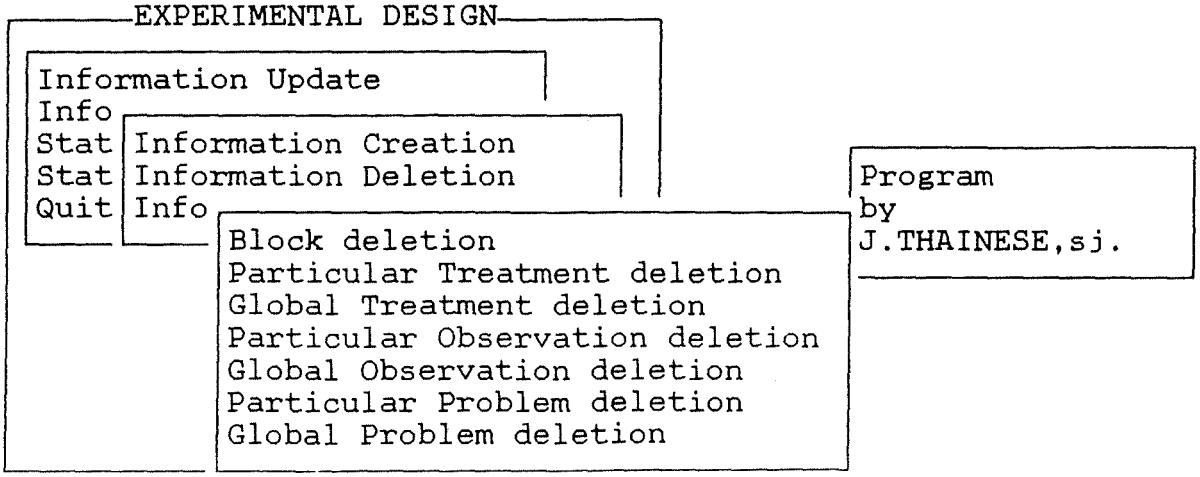
Press ESC key to go to the previous Menu

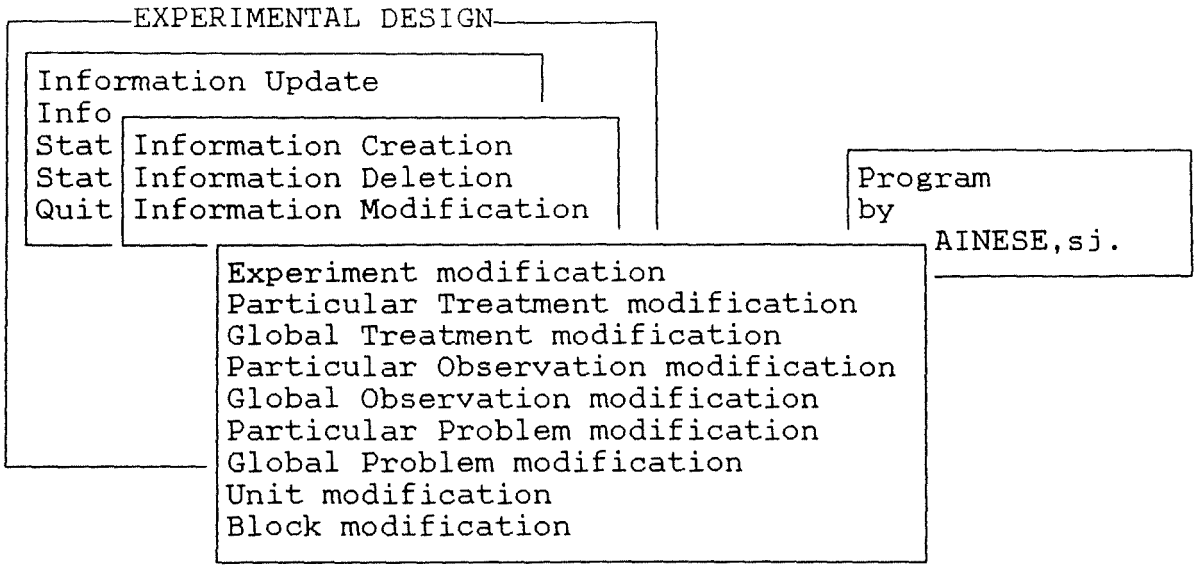


Press ESC key to go to the previous Menu

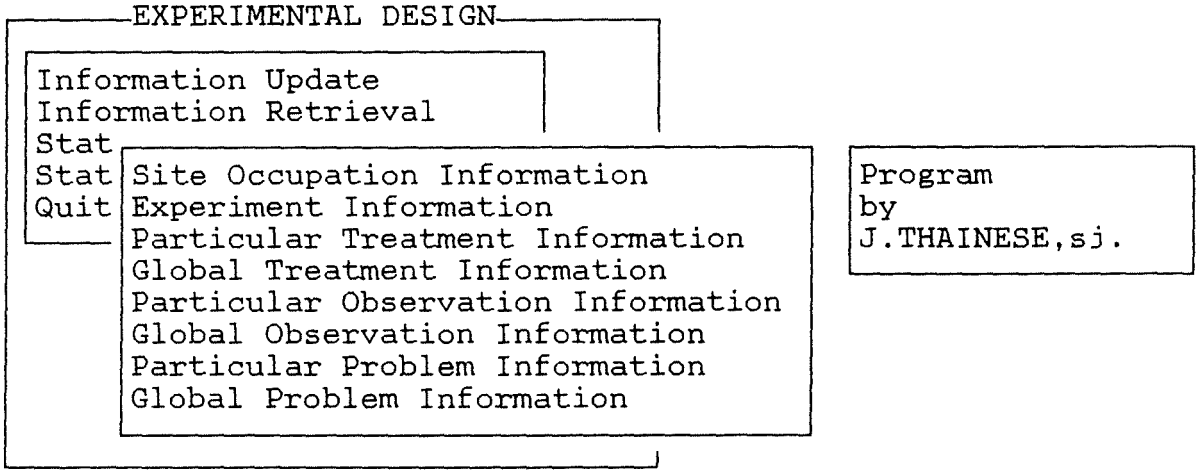


Press ESC key to go to the previous Menu

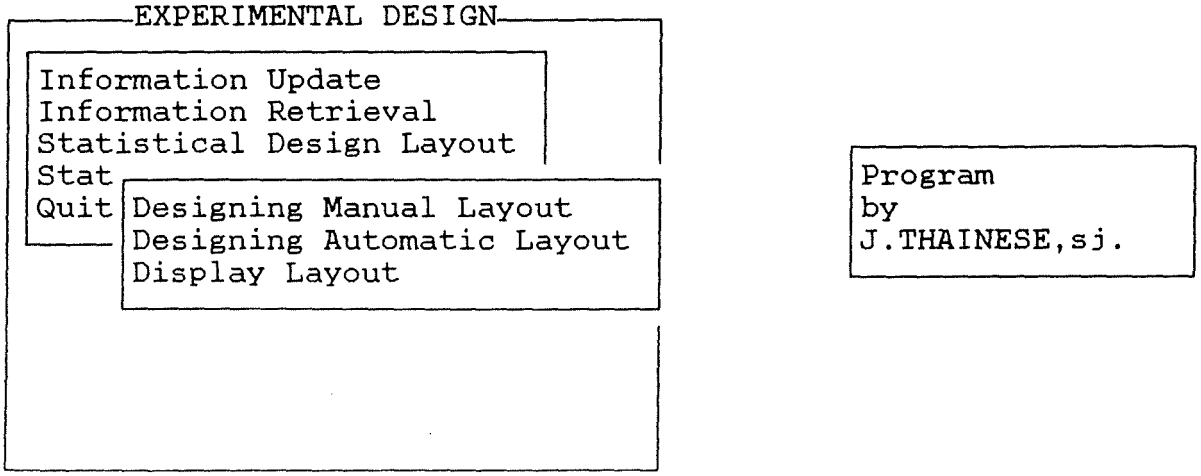


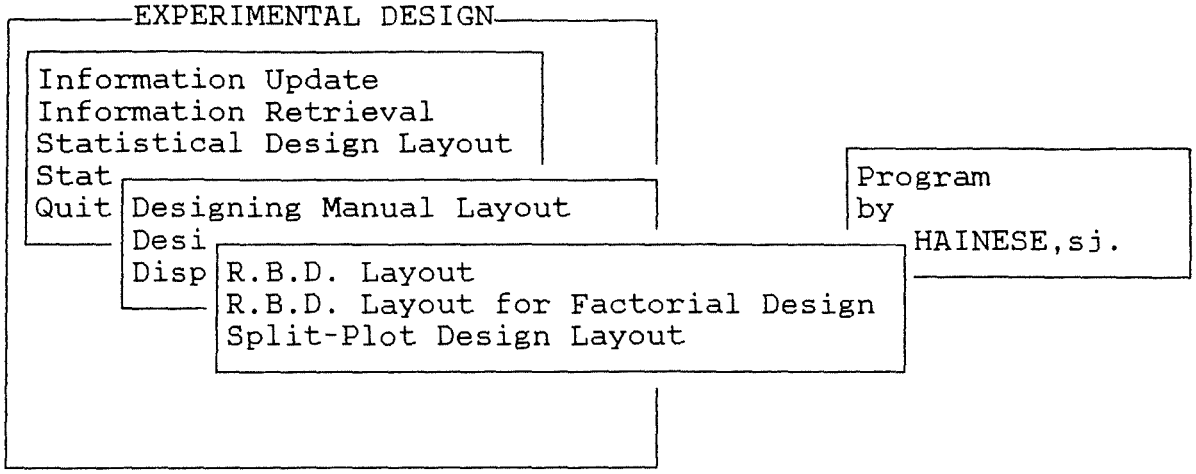


Press ESC key to go to the previous Menu

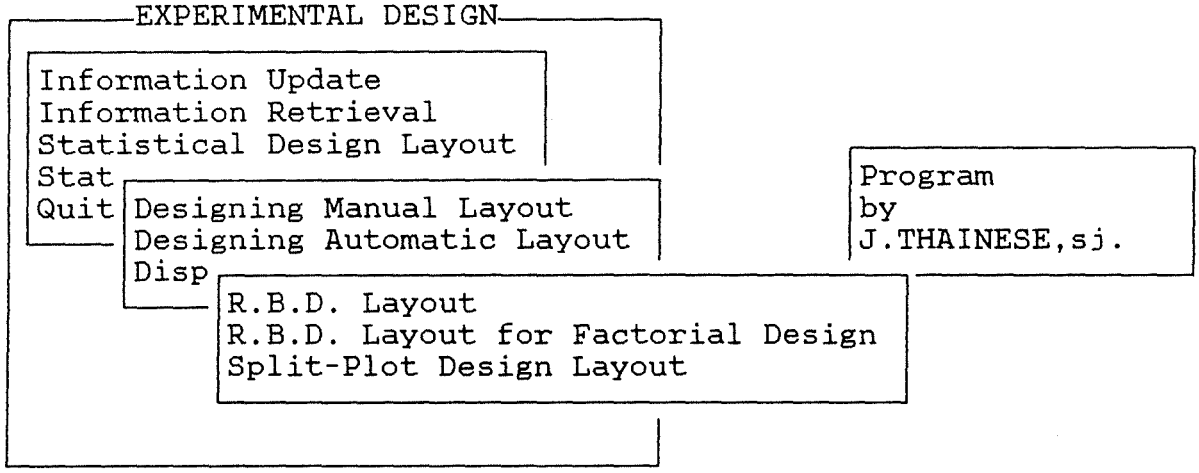


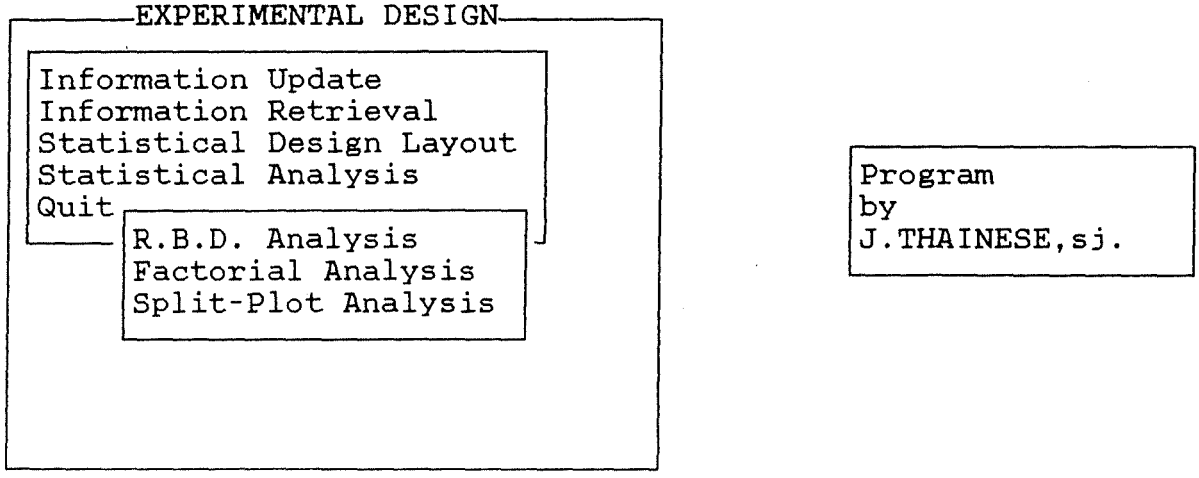
Press ESC key to go to the previous Menu





Press ESC key to go to the previous Menu





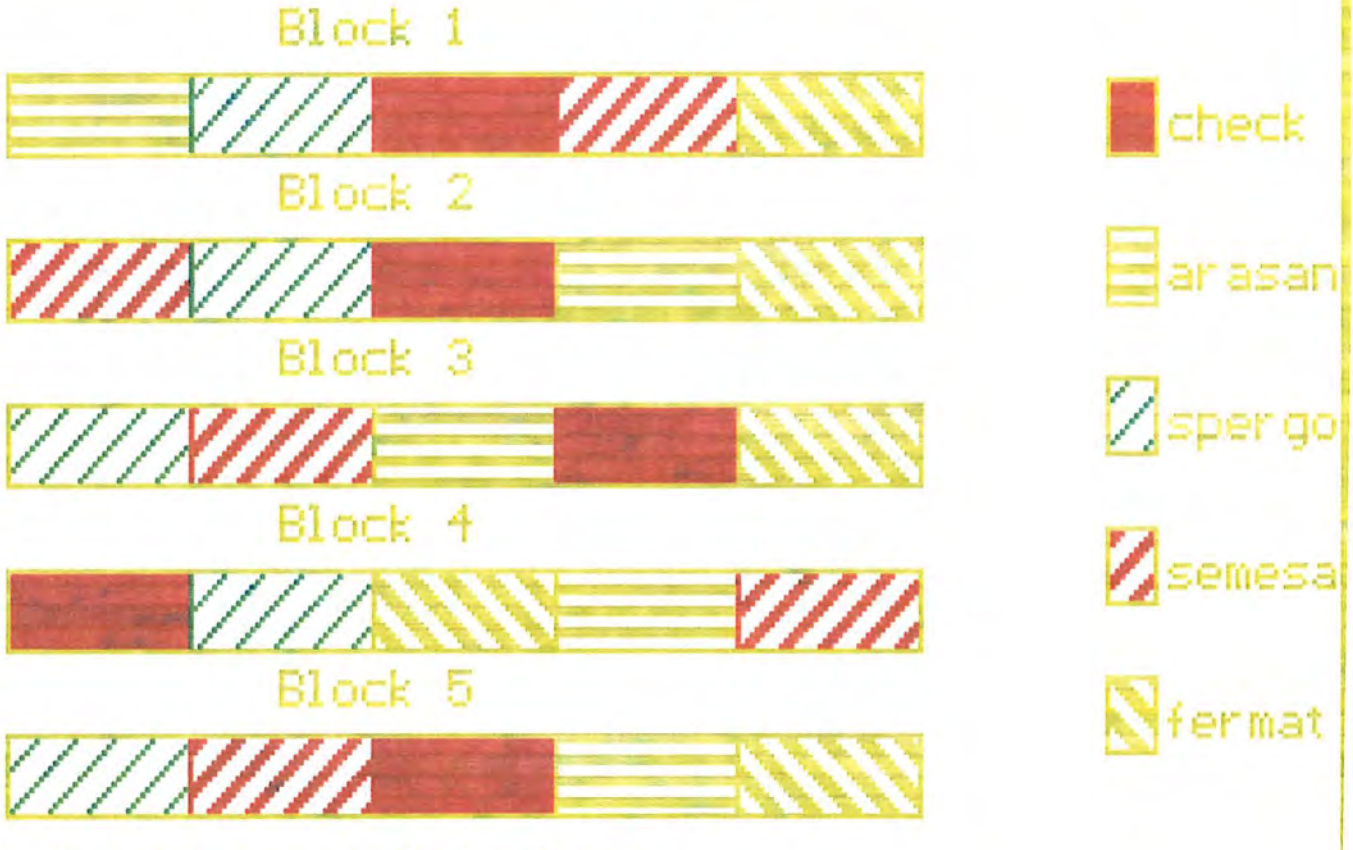
Press ESC key to go to the previous Menu

2. Data and Analysis for Randomized Block Design

The following example for Randomized Block Design is taken from the book by G.W. Snedecor and G. Cochran, *Statistical Methods* , Iowa University Press, Ames, Iowa, 1967 (sixth edition), p. 299-300, and adapted for our needs. Since the actual layout is not available, the function for designing automatic layout for Randomized Block Design is used. The data is collected from R.H. Porter, *Cooperative System Treatment Trials* , Iowa State College Seed Laboratory, 1936, in which four seed treatments were compared with no treatment (Check) on soyabean seeds. The data are the number of plants which failed to emerge out of 100 planted in each plot.

First we present the layout display and then the retreived data for analysis followed by a bar diagram for the yield of each block. Then, we present the sorted data for each block and for each treatment combination. Finally, the program displays a bar diagram for each treatment and the Analysis of Variance Table.

RANDOMIZED BLOCK EXP. NUMBER 5



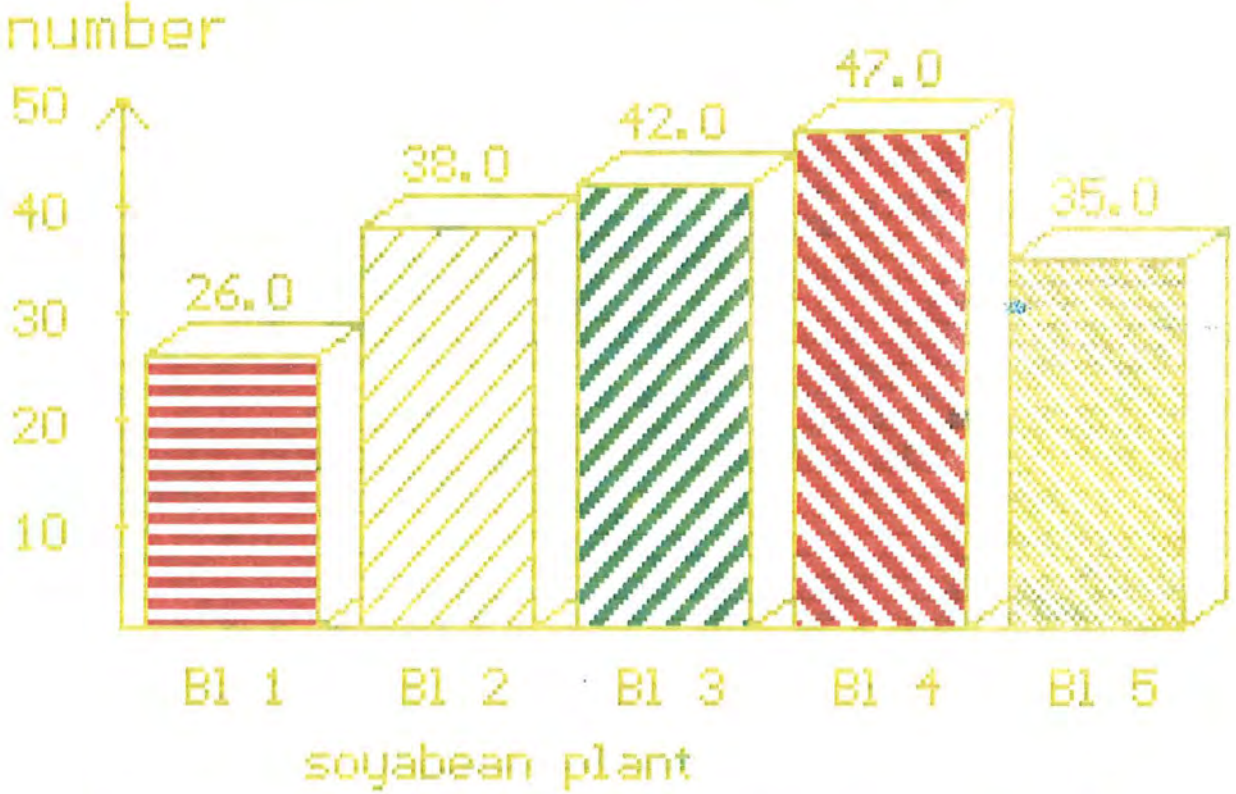
Press Esc to go to Main Menu

R.B.D. Analysis

```
Enter the object of the observation  :  soyabean plant count
unit qty # 1  :     2.000
unit qty # 2  :     4.000
unit qty # 3  :     8.000
unit qty # 4  :     3.000
unit qty # 5  :     9.000
block quantity #1  :    26.000

unit qty # 6  :     5.000
unit qty # 7  :    10.000
unit qty # 8  :    10.000
unit qty # 9  :     6.000
unit qty # 10  :     7.000
block quantity #2  :    38.000

unit qty # 11  :     9.000
unit qty # 12  :     9.000
unit qty # 13  :     7.000
unit qty # 14  :    12.000
unit qty # 15  :     5.000
block quantity #3  :    42.000

unit qty # 16  :    13.000
unit qty # 17  :     8.000
unit qty # 18  :     5.000
unit qty # 19  :    11.000
unit qty # 20  :    10.000
block quantity #4  :    47.000

unit qty # 21  :    10.000
unit qty # 22  :     6.000
unit qty # 23  :    11.000
unit qty # 24  :     5.000
unit qty # 25  :     3.000
block quantity #5  :    35.000
```

Press ESC key to go to the previous Menu

EXPERIMENT NUMBER 5



Press any key Missing Observations : 0

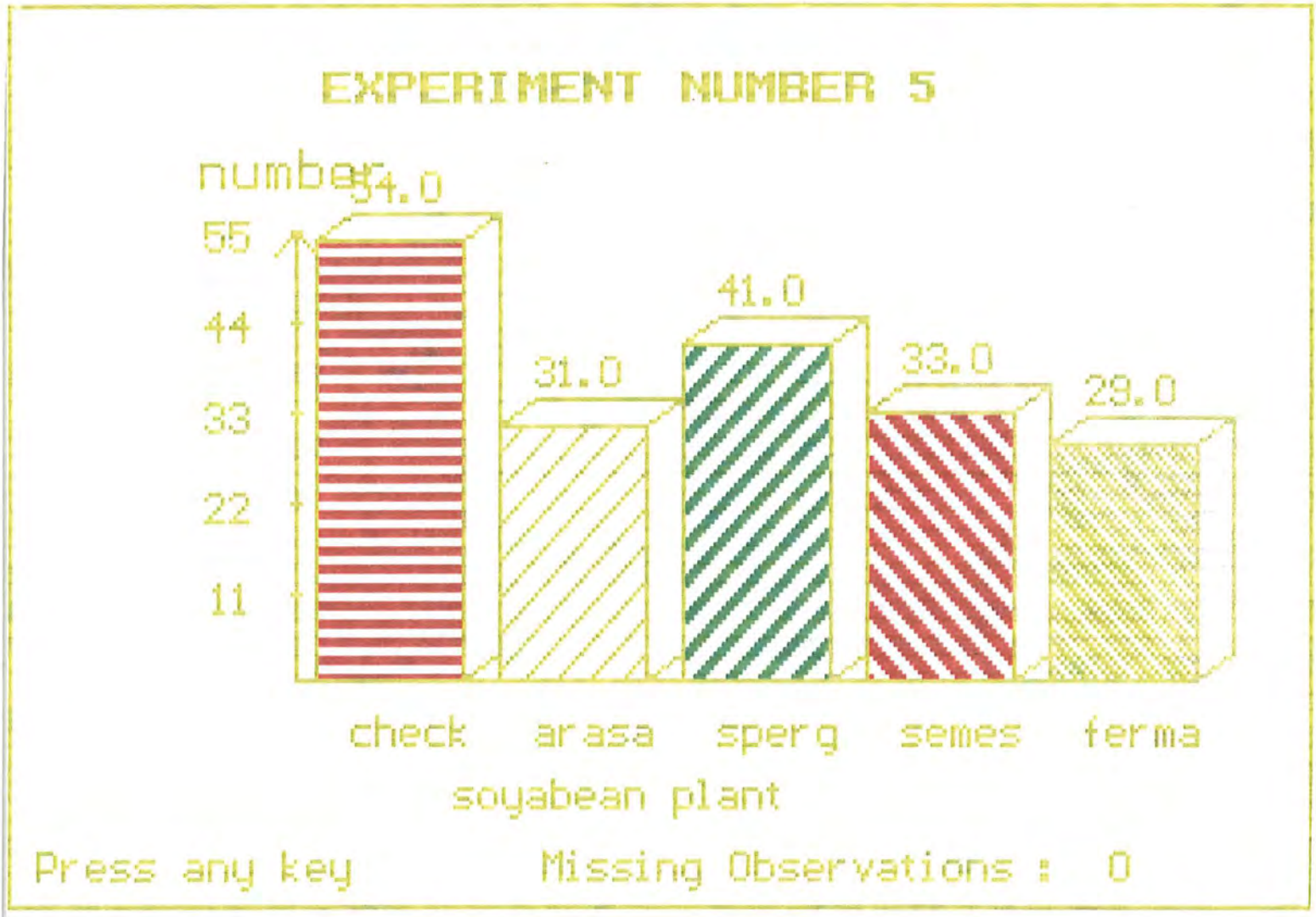
TREAT.COMB.	Bl 1	Bl 2	Bl 3	Bl 4	Bl 5	TOTAL
check	8.000	10.000	12.000	13.000	11.000	54.000
arasan	2.000	6.000	7.000	11.000	5.000	31.000
spergo	4.000	10.000	9.000	8.000	10.000	41.000
semesa	3.000	5.000	9.000	10.000	6.000	33.000
fermat	9.000	7.000	5.000	5.000	3.000	29.000
TOTAL	26.000	38.000	42.000	47.000	35.000	

press return key to go on.

ANALYSIS OF VARIANCE TABLE FOR RANDOMIZED BLOCK DESIGN

SOURCE	DF	SS	MEAN S S	F
BLOCK	4	49.8400	12.4600	2.3031
TREATMENT	4	83.8400	20.9600	3.8743
ERROR	16	86.5600	5.4100	
TOTAL	24	220.2400		

press return key and then Esc keyn

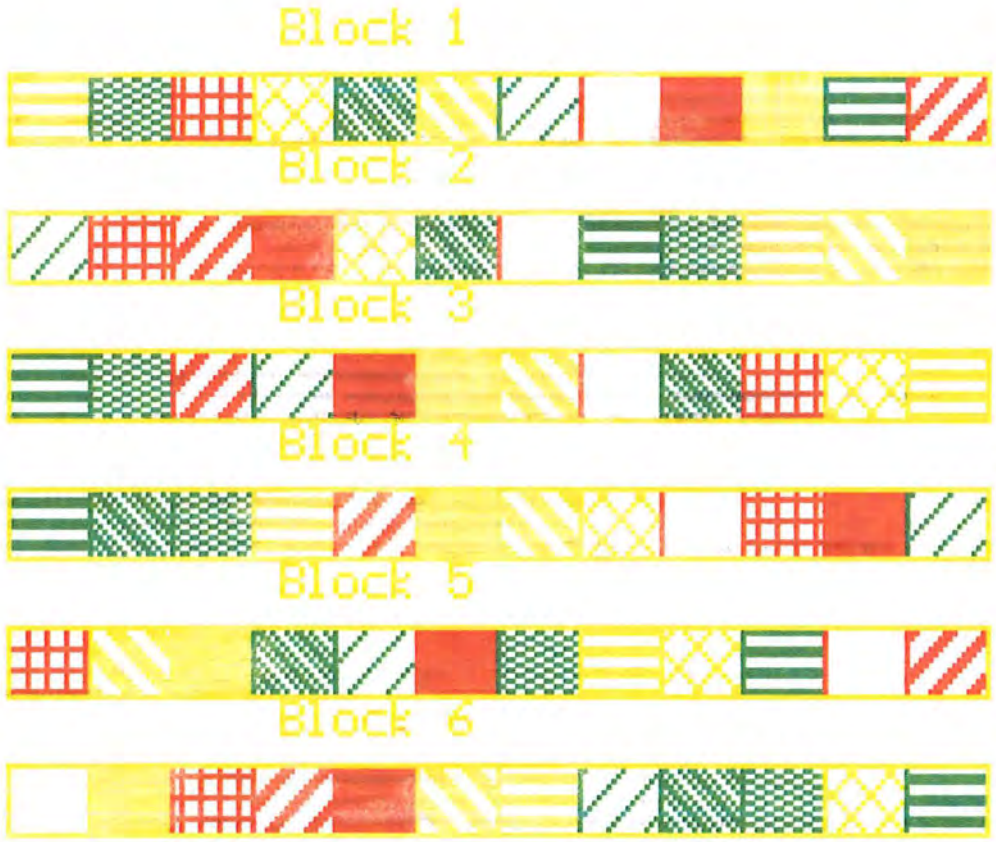

EXPERIMENT NUMBER 5
number
55
44
33
22
11
54.0
31.0
41.0
33.0
29.0
check
arasa
sperg
semes
ferma
soyabean plant
Press any key
Missing Observations : 0

3. Data and Analysis for 3x2x2 Mixed Factorial Design

The following example is complete with all the necessary data to be stored in the data base. It is taken from M.Sc. Project Report (cf. Krishnamoorty and J. Thainese, *Effect of Fertilizer and Spacing on Rice Yield* , Madras Christian College, India, 1977). Since the layout was already implemented, and all the data for it are available, the function for designing manual layout for Factorial Design is used to enter the data in the data base.

First we present the layout display and then the retreived data for analysis, followed by a bar diagram for the yield of each block and for each treatment. Then, the sorted data for each block and for each treatment combination is displayed. Finally, the Condensed Table and a bar diagram for each level of the factors and the Analysis of Variance Table are presented.

FACTORIAL DESIGN EXP.NUMBER 3



n1v1s1
n1v1s2
n1v2s1
n1v2s2
n2v1s1
n2v1s2
n2v2s1
n2v2s2
n3v1s1
n3v1s2
n3v2s1
n3v2s2

Press Esc to go to Main Menu

Factorial Analysis

```
Enter the object of the observation  :  paddy yield
unit qty # 1   :      3.348
unit qty # 2   :      2.966
unit qty # 3   :      3.475
unit qty # 4   :      2.953
unit qty # 5   :      2.515
unit qty # 6   :      2.779
unit qty # 7   :      3.385
unit qty # 8   :      2.597
unit qty # 9   :      3.012
unit qty # 10   :      2.928
unit qty # 11   :      3.300
unit qty # 12   :      3.583
block quantity #1   :     36.841
```

Press ESC key to go to the previous Menu

```
unit qty # 13   :      3.423
unit qty # 14   :      3.888
unit qty # 15   :      3.450
unit qty # 16   :      3.574
unit qty # 17   :      2.920
unit qty # 18   :      2.800
unit qty # 19   :      2.533
unit qty # 20   :      2.745
unit qty # 21   :      2.625
unit qty # 22   :      2.526
unit qty # 23   :      3.395
unit qty # 24   :      3.302
block quantity #2   :     37.181
```

Press ESC key to go to the previous Menu

```
unit qty # 25   :      3.220
unit qty # 26   :      3.262
unit qty # 27   :      3.160
unit qty # 28   :      3.480
unit qty # 29   :      3.595
unit qty # 30   :      3.435
unit qty # 31   :      3.197
unit qty # 32   :      2.613
unit qty # 33   :      3.150
unit qty # 34   :      3.050
unit qty # 35   :      3.407
unit qty # 36   :      3.195
block quantity #3   :     38.764
```

Press ESC key to go to the previous Menu

```
unit qty # 37  :      3.040
unit qty # 38  :      2.980
unit qty # 39  :      3.644
unit qty # 40  :      3.023
unit qty # 41  :      3.505
unit qty # 42  :      3.760
unit qty # 43  :      3.775
unit qty # 44  :      3.456
unit qty # 45  :      3.025
unit qty # 46  :      3.804
unit qty # 47  :      3.735
unit qty # 48  :      3.450
block quantity #4  :     41.197
```

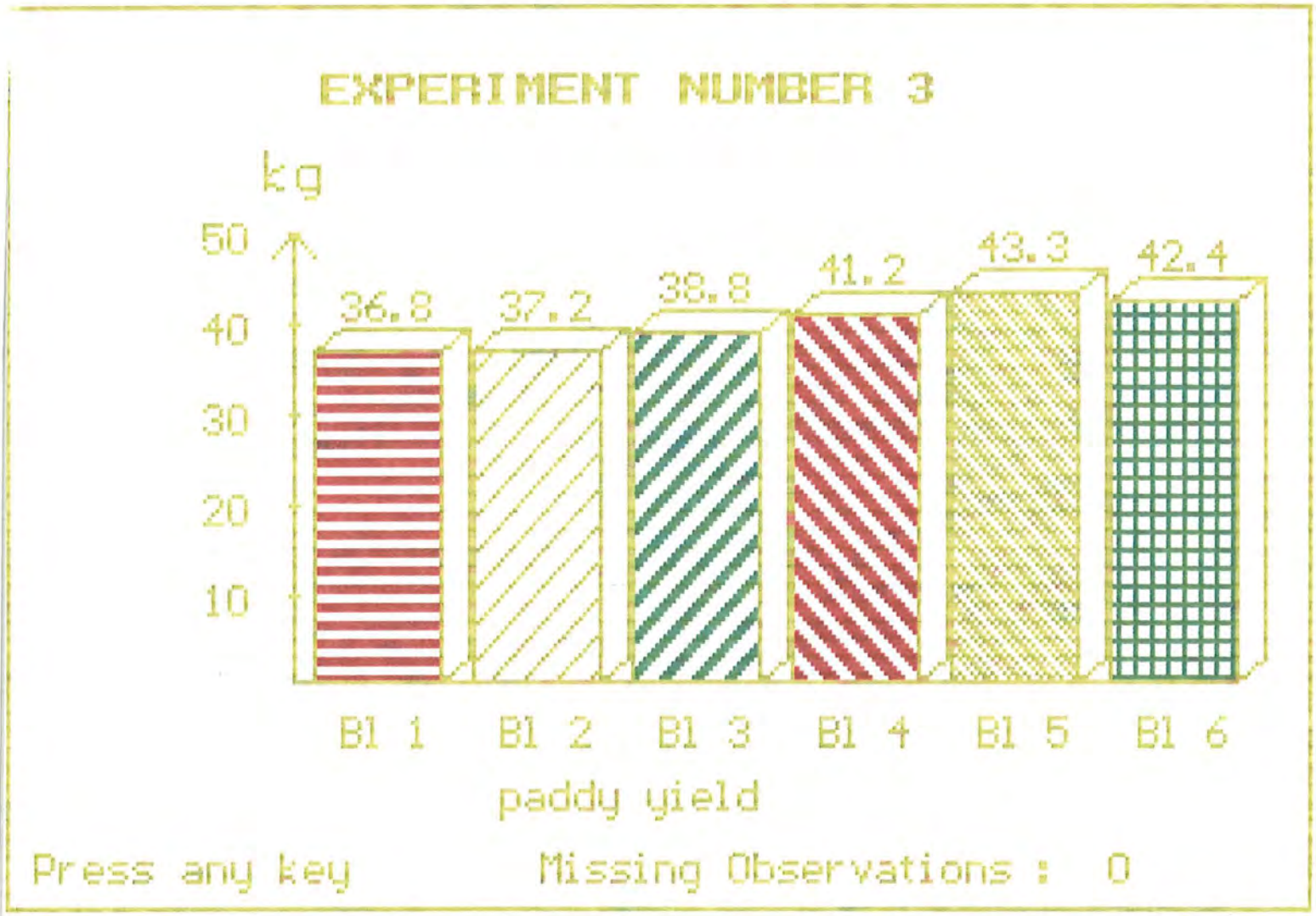
Press ESC key to go to the previous Menu

```
unit qty # 49  :      3.303
unit qty # 50  :      3.358
unit qty # 51  :      3.674
unit qty # 52  :      3.992
unit qty # 53  :      3.405
unit qty # 54  :      3.694
unit qty # 55  :      3.705
unit qty # 56  :      3.455
unit qty # 57  :      3.545
unit qty # 58  :      3.804
unit qty # 59  :      3.400
unit qty # 60  :      3.965
block quantity #5  :     43.300
```

Press ESC key to go to the previous Menu

```
unit qty # 61  :      3.252
unit qty # 62  :      3.862
unit qty # 63  :      3.862
unit qty # 64  :      3.290
unit qty # 65  :      3.390
unit qty # 66  :      3.645
unit qty # 67  :      3.080
unit qty # 68  :      3.664
unit qty # 69  :      3.698
unit qty # 70  :      3.490
unit qty # 71  :      3.630
unit qty # 72  :      3.502
block quantity #6  :     42.365
```

Press ESC key to go to the previous Menu


EXPERIMENT NUMBER 3
kg
50
40
30
20
10
36.8
37.2
38.8
41.2
43.3
42.4
Bl 1
Bl 2
Bl 3
Bl 4
Bl 5
Bl 6
paddy yield
Press any key
Missing Observations : 0

TREAT.COMB.	Bl 1	Bl 2	Bl 3	Bl 4	Bl 5	Bl 6	TOTAL
n1v1s1	3.012	3.574	3.595	3.735	3.694	3.390	21.000
n1v1s2	3.348	2.526	3.195	3.023	3.455	3.080	18.627
n1v2s1	3.385	3.423	3.480	3.450	3.405	3.664	20.807
n1v2s2	3.583	3.450	3.160	3.505	3.965	3.290	20.953
n2v1s1	2.779	3.395	3.197	3.775	3.358	3.645	20.149
n2v1s2	2.515	2.800	3.150	2.980	3.992	3.698	19.135
n2v2s1	3.475	3.888	3.050	3.804	3.303	3.862	21.382
n2v2s2	2.953	2.920	3.407	3.456	3.545	3.630	19.911
n3v1s1	2.966	2.625	3.262	3.644	3.705	3.490	19.692
n3v1s2	2.597	2.533	2.613	3.025	3.400	3.252	17.420
n3v2s1	2.928	3.302	3.435	3.760	3.674	3.862	20.961
n3v2s2	3.300	2.745	3.220	3.040	3.804	3.502	19.611
TOTAL	36.841	37.181	38.764	41.197	43.300	42.365	

press return key to go on.

	n 1	n 2	n 3
v 1	39.63	39.28	37.11
v 2	41.76	41.29	40.57

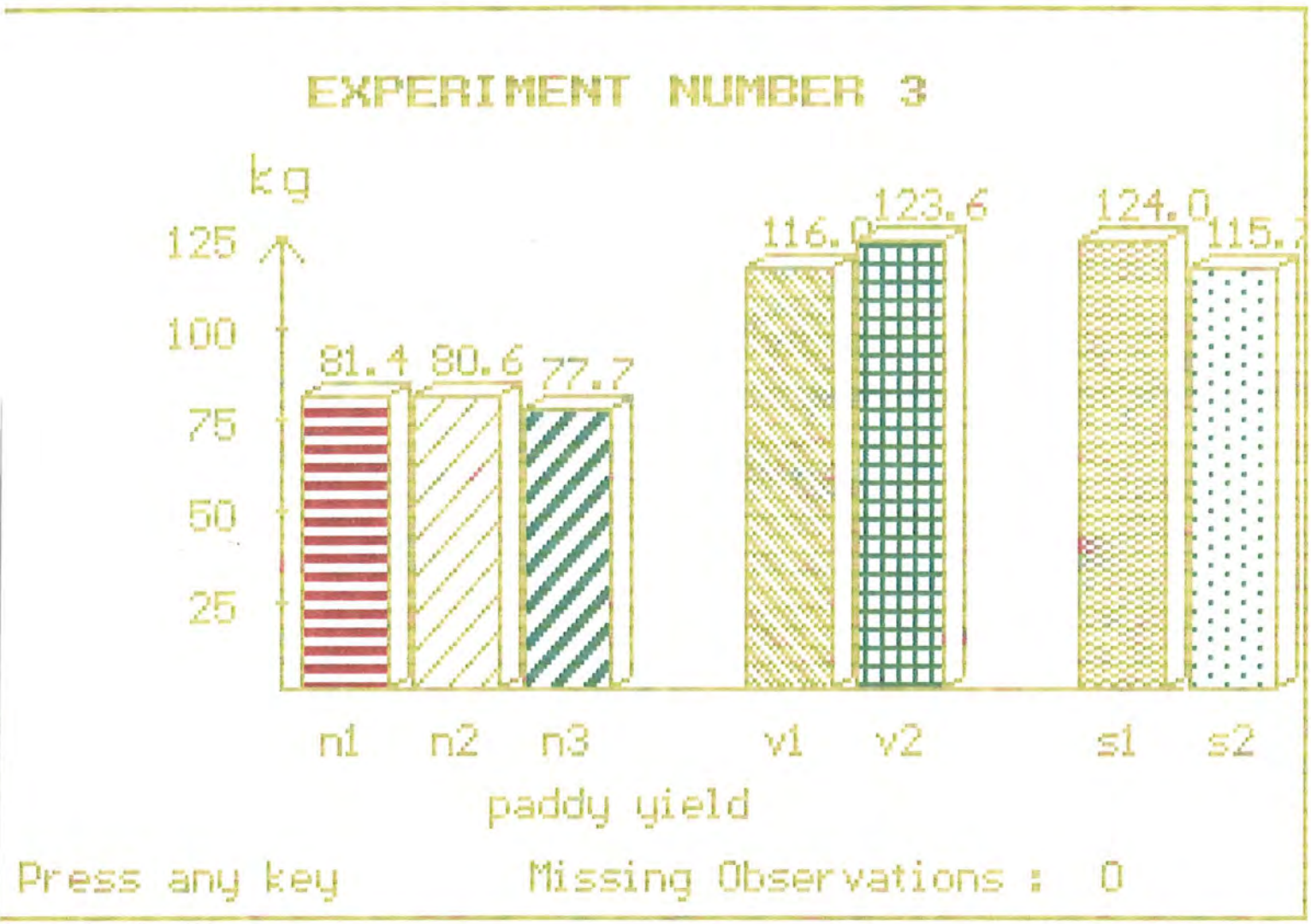
	n 1	n 2	n 3
s 1	41.81	41.53	40.65
s 2	39.58	39.05	37.03

	v 1	v 2
s 1	60.84	63.15
s 2	55.18	60.48

ANALYSIS OF VARIANCE TABLE FOR FACTORIAL DESIGN

SOURCE	DF	SS	MEAN S S	F
BLOCK	5	3.1124	0.6225	7.9614
TREATMENT	11	2.4914	0.2265	2.8968
nitrogen	2	0.3158	0.1579	2.0195
variety	1	0.8026	0.8026	10.2655
interaction nv	2	0.0539	0.0270	0.3448
space	1	0.9647	0.9647	12.3377
interaction vs	1	0.1237	0.1237	1.5817
interaction ns	2	0.0459	0.0230	0.2936
interaction nvs	2	0.1848	0.0924	1.1820
ERROR	55	4.3004	0.0782	
TOTAL	71	9.9042		

press return key and then Esc keyn

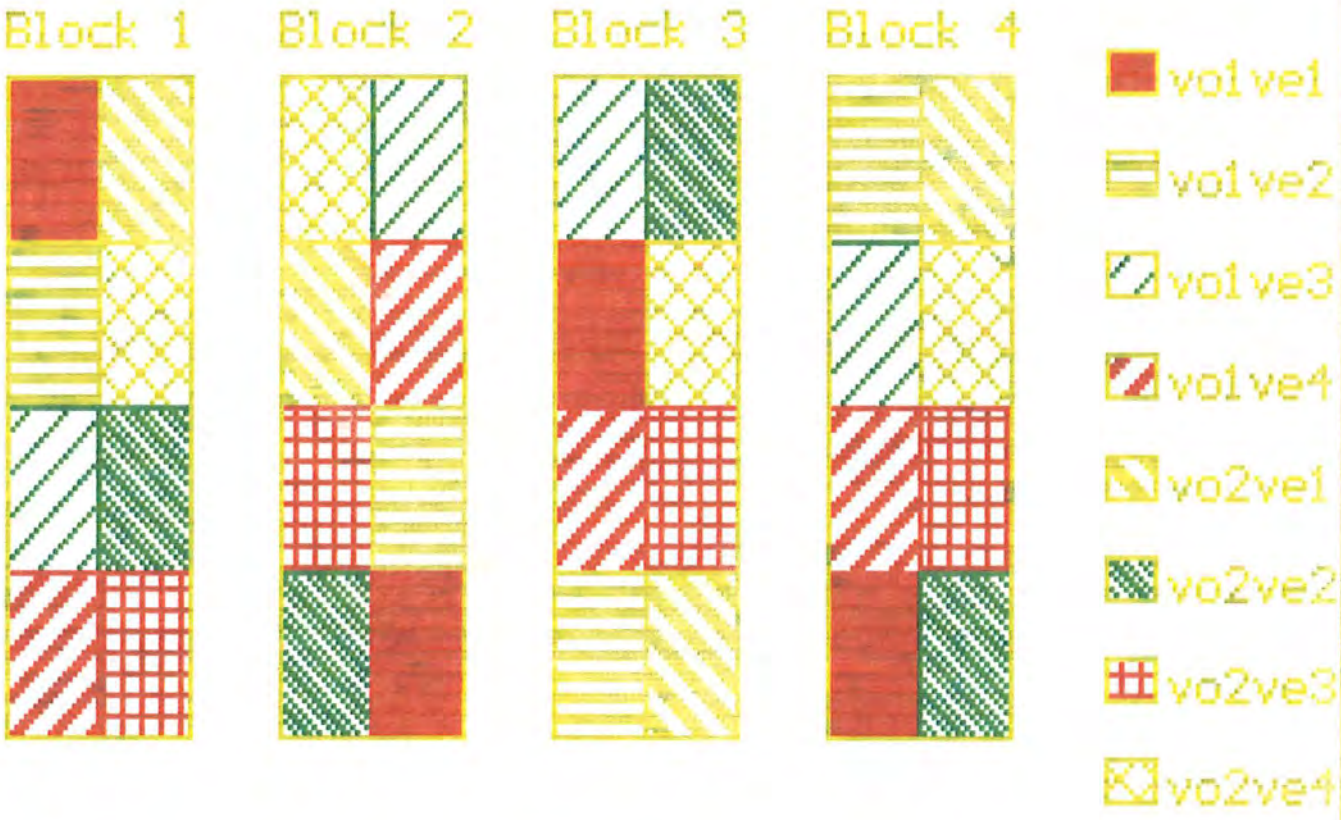

EXPERIMENT NUMBER 3
kg
125
100
75
50
25
81.4
80.6
77.7
116.0
123.6
124.0
115.2
n1
n2
n3
v1
v2
s1
s2
paddy yield
Press any key
Missing Observations : 0

4. Data and Analysis for Split-Plot Design

The example given in this section was received from the Faculty of Agronomical Sciences, Gembloux. The objective of this experiment is to compare the efficiency of a herbicide applied at different levels to cereals of different volumes. Since the actual layout was not available, the technique for automatic layout for Split-Plot design was employed.

First we present the layout display and then the retreived data for analysis, followed by a bar diagram for the yield of each block and for each treatment. Then, the sorted data for each block and for each treatment combination is displayed. Finally, the Condensed Table and a bar diagram for each level of the factors and the Analysis of Variance Table are presented.

SPLIT PLOT DESIGN EXP.NUMBER 4



Press Esc to go to Main Menu

Split-Plot Analysis

```
Enter the object of the observation  :  cereal yield
unit qty # 1   :  8118.000
unit qty # 2   :  8762.000
unit qty # 3   :  8700.000
unit qty # 4   :  8660.000
unit qty # 5   :  8704.000
unit qty # 6   :  8735.000
unit qty # 7   :  8968.000
unit qty # 8   :  9044.000
block quantity #1   :  69691.000
```

```
unit qty # 9   :  8512.000
unit qty # 10   :  8529.000
unit qty # 11   :  8617.000
unit qty # 12   :  8906.000
unit qty # 13   :  8995.000
unit qty # 14   :  8432.000
unit qty # 15   :  8534.000
unit qty # 16   :  8703.000
block quantity #2   :  69228.000
```

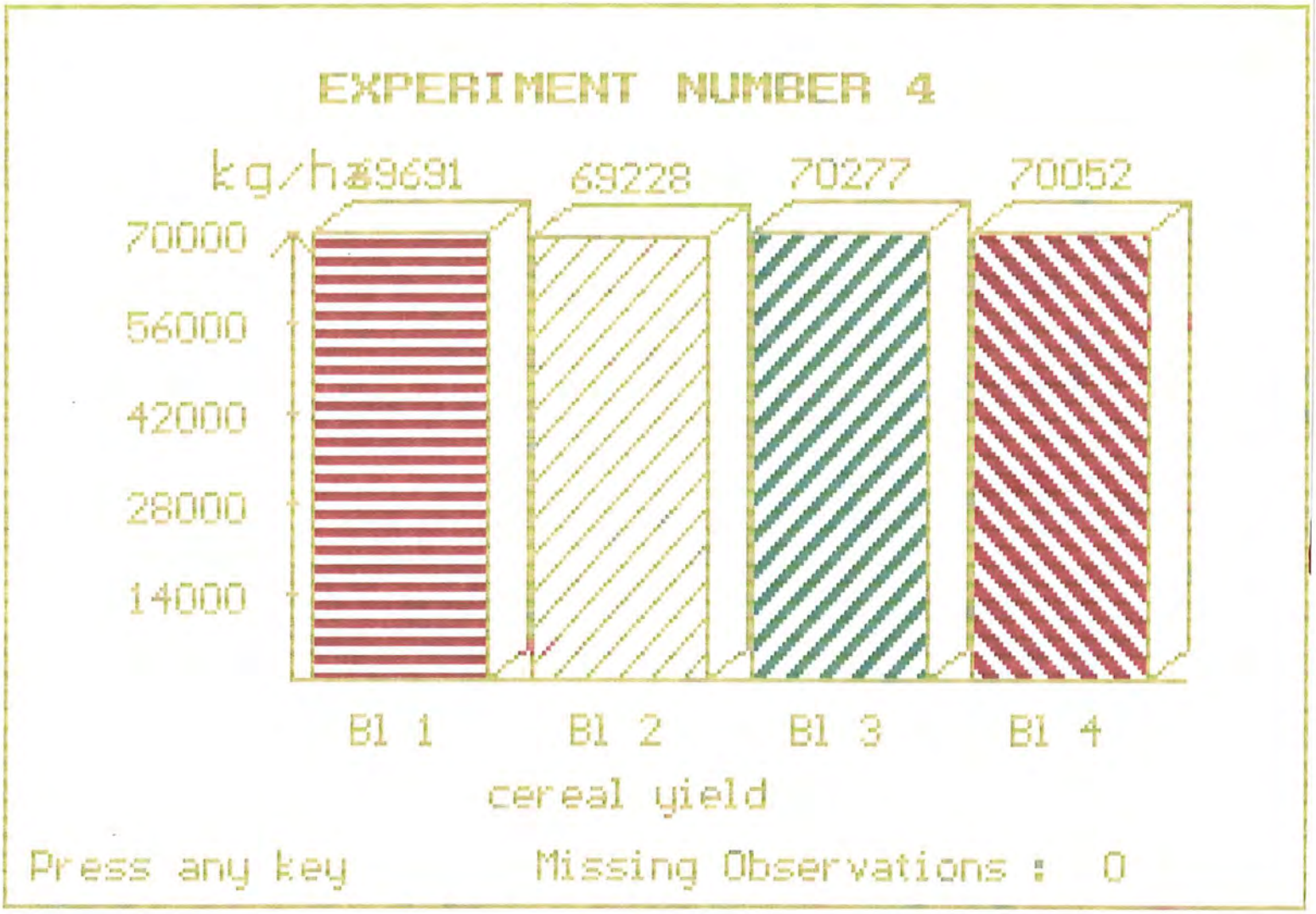
Press ESC key to go to the previous Menu

```
unit qty # 17   :  9374.000
unit qty # 18   :  8852.000
unit qty # 19   :  8531.000
unit qty # 20   :  8618.000
unit qty # 21   :  8970.000
unit qty # 22   :  8333.000
unit qty # 23   :  8899.000
unit qty # 24   :  8700.000
block quantity #3   :  70277.000
```

Press ESC key to go to the previous Menu

```
unit qty # 25   :  8842.000
unit qty # 26   :  8951.000
unit qty # 27   :  8911.000
unit qty # 28   :  8415.000
unit qty # 29   :  8454.000
unit qty # 30   :  8631.000
unit qty # 31   :  8766.000
unit qty # 32   :  9082.000
block quantity #4   :  70052.000
```

Press ESC key to go to the previous Menu


EXPERIMENT NUMBER 4
kg/hæ9691
69228
70277
70052
70000
56000
42000
28000
14000
Bl 1
Bl 2
Bl 3
Bl 4
cereal yield
Press any key
Missing Observations : 0

TREAT.COMB.	Bl 1	Bl 2	Bl 3	Bl 4	TOTAL
vo1ve1	8118.000	8703.000	8852.000	8415.000	34088.000
vo1ve2	8762.000	8534.000	8618.000	8842.000	34756.000
vo1ve3	8700.000	8995.000	9374.000	8951.000	36020.000
vo1ve4	8660.000	8432.000	8531.000	8911.000	34534.000
vo2ve1	8704.000	8529.000	8700.000	8454.000	34387.000
vo2ve2	8968.000	8906.000	8970.000	9082.000	35926.000
vo2ve3	9044.000	8617.000	8899.000	8766.000	35326.000
vo2ve4	8735.000	8512.000	8333.000	8631.000	34211.000
TOTAL	69691.000	69228.000	70277.000	70052.000	

press return key to go on.

	v 1	v 2
v 1	34088.00	34387.00
v 2	34756.00	35926.00
v 3	36020.00	35326.00
v 4	34534.00	34211.00

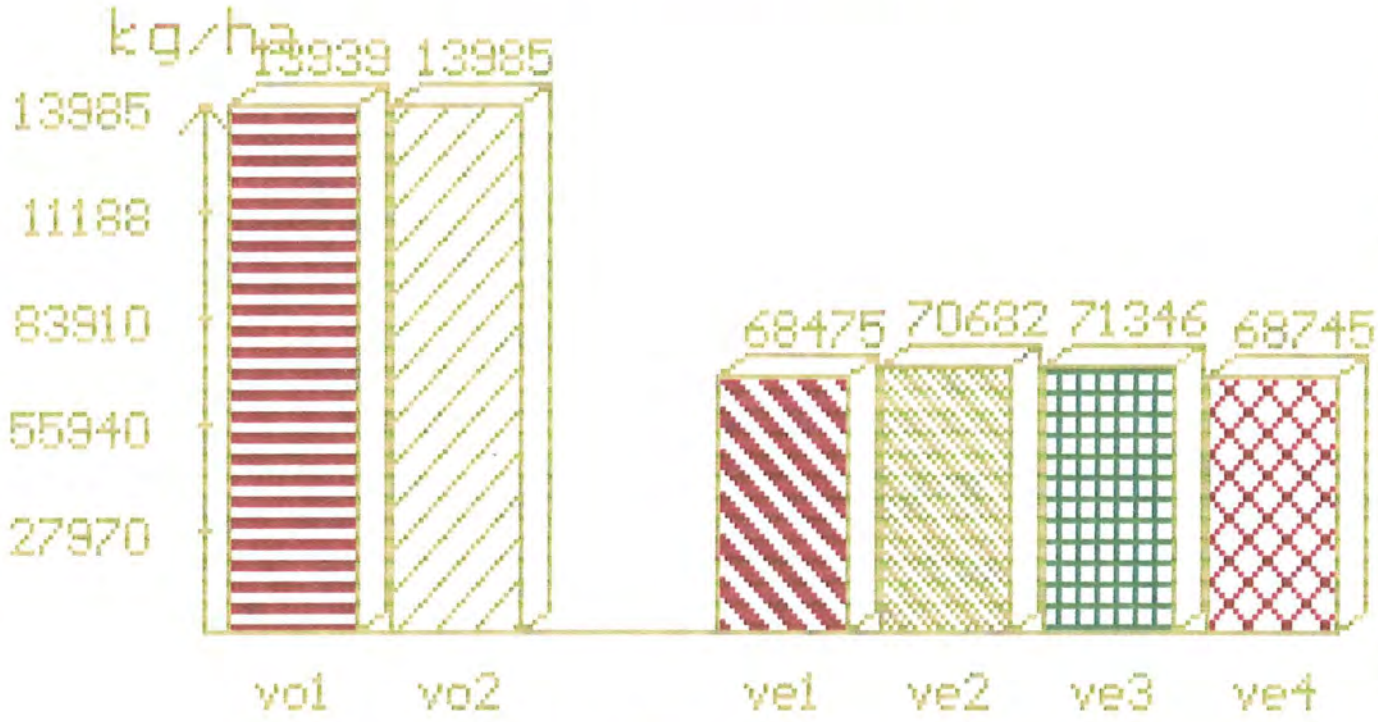
press return key to go on

ANALYSIS OF VARIANCE TABLE FOR SPLIT PLOT DESIGN

SOURCE	DF	SS	MEAN S S	F
BLOCK	3	78690.2500	26230.0833	0.6060
volume	1	6384.5000	6384.5000	0.1645
block x volume	3	210471.2500	70157.0833	1.8078
verigal	3	754514.2500	251504.7500	6.4808
interaction vv	3	249148.7500	83049.5833	2.1400
ERROR	18	698535.0000	38807.5000	
TOTAL	31	1997744.0000		
FOR block x volumeAS ERROR TERM				
volume	1	6384.5000	6384.5000	0.0910

press return key and then Esc keyn

EXPERIMENT NUMBER 4



Press any key

Missing Observations : 0